

UNITED STATES GOVERNMENT

Memorandum

TO : All 7040/7094 DCS Users

DATE: April 1, 1966

FROM : Computation & Analysis Branch

SUBJECT: Ames FORTRAN IV Subroutine Library Manual

1. A new subroutine AL PENR is available for line-plotting. It permits the EAI Dataplotters to be run at full speed. A 3:2 reduction in plotter time has been realized with the use of this subroutine. Write-ups are available from the Computation & Analysis Branch secretary.

2. CAB program numbers, for use with 7040/7094 DCS decks, will be assigned by the branch secretary, in Room 103. Instructions for filling out the white application form are in section 8.3.1 of the Ames 7040/7094 User's Manual.

3. It is anticipated that the 7040/7094 DCS will be replaced in Fiscal Year 1968. At that time it will no longer be possible to run FORTRAN II jobs. There remain a number of programs, some quite large, that have not been converted to FORTRAN IV. Users of FORTRAN II programs are urged to convert them to FORTRAN IV as soon as possible. The Computation & Analysis Branch staff is prepared to assist, particularly where FAP - MAP conversion is required.

4. The FORTRAN IV Library Tape now includes some 22 heavily used routines which were available, formerly, by requesting binary decks. The subroutines in the FORTRAN IV Library are available at load-time, and it is not necessary to request binary decks.

890K

N67-82034

FACILITY FORM 802	(ACCESSION NUMBER)	(THRU)
	10 167 1500	None
	(PAGES)	(CODE)
	(NASA CR OR TMX OR AD NUMBER)	(CATEGORY)
	OR 80086 IND	
	TM-X-59464	

1 N 11. Ames 3

TABLE I

SHARE CATALOG CLASSIFICATIONS

Programs shall be assigned a 2-character classification code. The leftmost character is a letter indicating a primary class; the second character is a digit indicating a secondary class within the primary. The classifications shall be as follows:

A. Arithmetic Routines

- | | |
|-------------------|---------------------------------------------------------------------|
| 1. Real Numbers | May include multiple precision, fixed and floating-point operations |
| 2. Complex Number | May include multiple precision, fixed and floating-point operations |
| 3. Decimal | BCD single or multiple precision arithmetic operations |

B. Elementary Functions

- | | |
|--------------------------------|--------------------------------------------------|
| 1. Trigonometric | Also pertains to inverse trigonometric functions |
| 2. Hyperbolic | |
| 3. Exponential and Logarithmic | |
| 4. Roots and Powers | Refers to roots of quantities, not polynomials |

C. Polynomials and Special Functions

1. Evaluation of Polynomials
2. Roots of Polynomials
3. Evaluation of Special Functions
4. Simultaneous Non-linear Algebraic Equations
5. Simultaneous Transcendental Equations

D. Operations on Functions and Solutions of Differential Equations

1. Numerical Integration
2. Numerical Solutions of Ordinary Differential Equations
3. Numerical Solutions of Partial Differential Equations
4. Numerical Differentiation

E. Interpolation and Approximations

1. Table Look-up and Interpolation
2. Curve Fitting
3. Smoothing

F. Operations on Matrices, Vectors and Simultaneous Linear Equations

1. Matrix Operations
2. Eigenvalues and Eigenvectors
3. Determinants
4. Simultaneous Linear Equations

G. Statistical Analysis and Probability

1. Data Reduction Refers to the calculation of the more common statistical parameters such as mean, median, standard deviation, etc.
2. Correlation and Regression Analysis Includes curve fitting which is explicitly for statistical purposes
3. Sequential Analysis
4. Analysis of Variance

H. Operations Research, Linear Programming, Simulation, Scientific Management, Gaming and Game-Like Models

1. Linear Programming
2. General and Job-Shop Simulators
3. Games and Game-like Models
4. Game Theory
5. General Problem Solvers
6. Schedulers and Scientific Management

I. Input

1. Binary Pertains to program input or data input in the binary mode (via card, tape, drum or disk)
2. Octal Pertains to program input or data input in actual mode (cards)
3. Decimal Pertains to program input and data input in the decimal mode (via card or tape)

I. Input (continued)

- | | |
|-------------------------|---------------------------------------------------------------------------------------------------------------------|
| 4. BCD (Hollerith) | Pertains to program input or data input in the BCD or Hollerith mode (via card, tape, drum or disk) |
| 5. (formerly composite) | Reserved. Please do not use. |
| 9. Composite | A combination of any of the above, which is not primarily one of the above, such as a general purpose input program |

J. Output

- | | |
|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Binary | Pertains to program output (card or tape) or data output (card, tape, drum or disk) in the binary mode |
| 2. Octal | Pertains to program output (printer) or data output (card or printer) in the octal mode |
| 3. Decimal | Pertains to program output (card, tape or printer) or data output (card, tape or printer) in the decimal mode |
| 4. BCD (Hollerith) | Pertains to program output (card, tape, printer, drum or disk) or data output (card, tape, printer, drum or disk) in the BCD mode |
| 5. Plotting | Refers to routines for producing plotted output, either via printer or via CRT, or other special plotting device. Routines for using plotting devices to simulate printing are also included |
| 6. (formerly Plotting) | Reserved, Please do not use |
| 9. Composite | A combination of any of the above, which is not primarily one of the above, such as a general purpose output program |

K. Internal Information Transfer

Generally denotes core-to-core, tape-to-tape, drum-to-drum and core-to-tape and core-to-drum movements

K. Internal Information Transfer (continued)

- | | |
|------------------------|------------------------------------------------------------------------------------------------|
| 1. Drum | Any drum read/write, editing, duplicating or comparing, etc. program |
| 2. Relocation | Pertains to core-to-core or drum to-drum relocation only, not input with relocation |
| 3. Disk | Pertains to disk-to-disk, core-to-disk, disk-to-core, tape-to-disk and disk-to-tape relocation |
| 4. Tape | Any tape read/write, editing, duplicating or comparing, etc. program |
| 5. Direct Data Devices | Computer-to-computer information transfer, other than via the above categories |

L. Executive Routines

1. Assembly
2. Compiling
3. Monitoring
4. Preprocessing
5. Disassembly and De-Relativizing
6. Relativizing
7. Computer Language to Computer Language Translators

This refers to translation from one artificial language designed for computing and data processing purposes to another such language, e.g. FORTRAN to COBOL. Not to be used for translation of natural languages such as English or Russian.

M. Data Handling

- | | |
|------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. Sorting | |
| 2. Conversion and/or Scaling | Pertains to any conversion and scaling routine (packed or unpacked, single or multiple precision) such as card image to BCD, BCD to card image, binary to BCD, BCD to binary, fixed to floating, etc. The primary function of programs in this category must be conversion or scaling, not input-output. |

M. Data Handling (continued)

- 3. Merging
- 4. Character Manipulation

N. Debugging

- 1. Tracing: Trapping
- 2. Dumping Core tape, drum, disk, console
 printouts (on-or off-line)
- 3. Memory Verification
 and Searching
- 4. Breakpoint Printing

O. Simulation of Computers and Data Processors; Interpreters

- 1. Off-line Equipment Any program which simulates off-
 line equipment
- 2. Reserved. Please do not use.
- 3. Computers Pertains to programs which simulate
 or interpret other computers on
 the 704, 709, 7090 or successors
- 4. Pseudo-computers Simulation of theoretical or
 pseudo-computers
- 9. Other or composite

- P. Diagnostics Pertains to any program which
 checks for malfunctioning of the
 computer or its components

Q. Service or Housekeeping; Programming Aids

Pertains to any routine of a
utilitarian nature which performs
a service for the programmer such
as executing the equivalent of
pushing a button on the console,
setting a dial or accumulating
a check sum.

- 1. Clear/Reset Programs
- 2. Check Sum Accumulation and Correction

Q. Service or Housekeeping; Programming Aids (continued)

3. Rewind, Tape Mark, Load Cards, Load Tape, etc. Programs
4. Internal Housekeeping: Save, Restore, etc.
5. Report Generator Subroutines

R. Logical and Symbolic

Logical functions, logical operations, logical calculuses and algebras, symbol manipulation and manipulation of non-numeric quantities.

1. Formal Logic
2. Symbol Manipulation

S. Information Retrieval

T. Applications and Application-Oriented Programs

1. Physics (including nuclear)
2. Chemistry
3. Other Physical Sciences (Geology, Astronomy, etc.)
4. Engineering
5. Business Data Processing
6. Manufacturing, (non-data) Processing, and Process Control
7. Mathematics and Applied Mathematics

U. Linguistics and Languages

V. General Purpose Utility Subroutines

1. Random number Generators
2. Combinatorial Generators Permutations, Combinations, and Subsets

Z. All Others

This category contains all routines for which no primary class has yet been designated. Routines which are covered by a primary class but which are not adequately described by a sub-class are assigned the applicable primary classification with a sub-class designation of zero.

The SHARE classifications shall be reviewed from time to time by a committee appointed from the SHARE Membership. It is anticipated that new program developments will show the need for additional

classifications. Any program for which no suitable secondary class exists may be assigned the secondary code 0 (zero). The committee may assign additional secondary classes as required to distribute the accumulation of "0" classed programs into applicable classes.

Similarly, where no suitable primary class exists, a program may be classed "Z". The committee may establish additional primary classes as required.

The establishment of new classification codes is reserved for the committee. Suggestions for additional classifications may be submitted to the committee by the membership.

*ACOS*AMES*LIBRARY*INVERSE COSINE OR*ARCCOSINE
*AITKEN METHOD*AMES*LIBRARY*TABLE LOOK-UP AND*INTERPOLATION BY
*ALOG*AMES*LIBRARY*NATURAL*LOGARITHM OF A NUMBER
*ALOG10*AMES*LIBRARY*COMMON*LOGARITHM OF A NUMBER
*AMES*LIBRARY COMPUTE*SCALE FACTORS FOR*PLOTS
*AMES*LIBRARY DUMMY*PLOT PROGRAM
*AMES*LIBRARY DUMMY*PLOT PROGRAM
*AMES*LIBRARY EVALUATION OF A*POLYNOMIAL
*AMES*LIBRARY PRINT*SCALES ON*PLOTS GENERATED BY*EAI PLOTTER
*AMES*LIBRARY SELECTIVE*DUMP PROGRAM*PDUMP
*AMES*LIBRARY UTILITY PROGRAM
*AMES*LIBRARY UTILITY PROGRAM FOR*ERROR MESSAGES
*AMES*LIBRARY UTILITY PROGRAM FOR POSITIONING*TAPES*LOCATE
*AMES*LIBRARY UTILITY PROGRAM FOR*TITLE AND*DATE ON OUTPUT*PAGE
*AMES*LIBRARY UTILITY PROGRAM TO*REWIND AND UNLOAD MAG*TAPE
*AMES*LIBRARY*ARDC MODEL*ATMOSPHERE OF 1959
*AMES*LIBRARY*BOOLEAN OPERATION*AND OR*INTERSECTION
*AMES*LIBRARY*BOOLEAN OPERATION*COMPLEMENT OR*NOT
*AMES*LIBRARY*BOOLEAN OPERATION*OR OR*UNION
*AMES*LIBRARY*CLOCK ACCESS PROGRAM*CLOCK
*AMES*LIBRARY*COMMON*LOGARITHM OF A NUMBER*ALOG10
*AMES*LIBRARY*COSINE OF AN ANGLE
*AMES*LIBRARY*DATA CONVERSION BY ARBITRARY*FORMAT USE
*AMES*LIBRARY*DEFINE FULL LOGICAL WORD*BOOLEAN VARIABLE
*AMES*LIBRARY*DUMP COUNTING ROUTINE*COUNT
*AMES*LIBRARY*DUMP ROUTINE
*AMES*LIBRARY*DYNAMIC*DUMP PROGRAM*DDUMP
*AMES*LIBRARY*EXPONENTIAL FUNCTION
*AMES*LIBRARY*FLOATING POINT*OVERFLOW/UNDERFLOW PROGRAM
*AMES*LIBRARY*HYPERBOLIC COSINE*COSH FUNCTION
*AMES*LIBRARY*HYPERBOLIC SINE*SINH FUNCTION
*AMES*LIBRARY*HYPERBOLIC TANGENT*TANH FUNCTION
*AMES*LIBRARY*INVERSE COSINE OR*ARCCOSINE*ACOS
*AMES*LIBRARY*INVERSE TANGENT OR*ARCTANGENT*ARTN
*AMES*LIBRARY*INVERSE SINE OR*ARCSINE*ASIN
*AMES*LIBRARY*INVERSE TANGENT OR*ARCTANGENT*ATAN
*AMES*LIBRARY*INVERSE TANGENT OR*ARCTANGENT*ATAN2
*AMES*LIBRARY*LEAST SQUARES*POLYNOMIAL*CURVE FITTING
*AMES*LIBRARY*LINEAR*SIMULTANEOUS EQUATIONS BY*CROUT METHOD
*AMES*LIBRARY*MATRIX INVERSION AND*SIMULTANEOUS EQN SOLUTION

AL ACOS
AL TAIN
IBM ALOG
IBM ALG1
AL SCAL
AL PLOT
AL PSCA
AL POLY
AL SFAC
IBM PDMP
AL EDFL
AL EROR
AL LOCA
AL PAGE
AL REWN
AL ARDC
IBM AND
IBM CMPL
IBM OR
AL CLOK
IBM ALG1
IBM COS
AL CVRT
IBM BOOL
AL COUN
IBM DUMP
AL DDMP
IBM EXP
AL FPT
AL COSH
AL SINH
IBM TANH
AL ACOS
AL ARTN
AL ASIN
IBM ATAN
IBM ATN2
AL LSQP
AL CROT
AL MTNV

*AMES*LIBRARY*MATRIX INVERSION AND*SIMULTANEOUS EQUATIONS SOLUT
*AMES*LIBRARY*MATRIX INVERSION AND*SIMULTANEOUS LINEAR EQUATION
*AMES*LIBRARY*NATURAL*LOGARITHM OF A NUMBER*ALOG
*AMES*LIBRARY*NUMERICAL SOLUTION OF*DIFFERENTIAL EQUATIONS
*AMES*LIBRARY*NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS
*AMES*LIBRARY*NUMERICAL INTEGRATION OF A FUNCTION*GAUSS QUAD.
*AMES*LIBRARY*NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS
*AMES*LIBRARY*NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS
*AMES*LIBRARY*NUMERICAL*INTEGRATION BY*SIMPSON'S RULE
*AMES*LIBRARY*PARITY OF AN INTEGER*XPAP
*AMES*LIBRARY*PLOT*LABELS
*AMES*LIBRARY*PLOT*LABELS
*AMES*LIBRARY*PLOTWS PREPARES TAPE FOR*PLOTS ON*EAI XY PLOTTER
*AMES*LIBRARY*POST MORTEM*DUMP ROUTINE*CRASH
*AMES*LIBRARY*RANDOM NUMBER GENERATOR
*AMES*LIBRARY*RANDOM NUMBER GENERATOR
*AMES*LIBRARY*RANDOM NUMBER GENERATOR
*AMES*LIBRARY*READ*BINAR*Y*TAPE
*AMES*LIBRARY*REWIND*BACKSPACE*END-OF-FILE MARK ON*MAGNETIC TAP
*AMES*LIBRARY*ROOT OF FUNCTION BY*INTERVAL HALVING METHOD
*AMES*LIBRARY*ROOT OF FUNCTION BY*INTERVAL HALVING METHOD
*AMES*LIBRARY*ROOT OF AN ARBITRARY FUNCTION*NEWTON-RAPHSON
*AMES*LIBRARY*ROOTS OF A*POLYNOMIAL*DOUBLE PRECISION*REAL COEF
*AMES*LIBRARY*ROOTS OF A*POLYNOMIAL
*AMES*LIBRARY*ROOTS OF*POLYNOMIALS WITH REAL COEFFICIENTS
*AMES*LIBRARY*SIMULTANEOUS*LINEAR EQUATIONS
*AMES*LIBRARY*SINE OF AN ANGLE
*AMES*LIBRARY*SMOOTH AND*DIFFERENTIATE SET OF DATA POINTS
*AMES*LIBRARY*SQUARE ROOT FUNCTION
*AMES*LIBRARY*TABLE LOOK-UP AND*INTERPOLATION BY*AITKEN METHOD
*AMES*LIBRARY*TANGENT FUNCTION
*AND OR*INTERSECTION*AMES*LIBRARY*BOOLEAN OPERATION
*ARCCOSINE*ACOS*AMES*LIBRARY*INVERSE COSINE OR
*ARCSINE*ASIN*AMES*LIBRARY*INVERSE SINE OR
*ARCTANGENT*ARTN*AMES*LIBRARY*INVERSE TANGENT OR
*ARCTANGENT*ATAN*AMES*LIBRARY*INVERSE TANGENT OR
*ARCTANGENT*ATAN2*AMES*LIBRARY*INVERSE TANGENT OR
*ARDC MODEL*ATMOSPHERE OF 1959*AMES*LIBRARY
*ARTN*AMES*LIBRARY*INVERSE TANGENT OR*ARCTANGENT
*ASIN*AMES*LIBRARY*INVERSE SINE OR*ARCSINE

AL INVT
AL MINV
IBM ALOG
AL ADAM
AL COWL
AL GAUS
AL INT
AL DE6F
AL SIMP
AL XPAP
AL ALPP
AL PALP
AL PLTW
AL CRSH
AL BARN
AL RNDM
AL RDM
AL BNIN
AL NDID
AL ITR2
AL ITRA
AL ROOT
AL DPMU
AL ROP
ML HPRS
AL LSQS
IBM SIN
AL DIF3
IBM SQRT
AL TAIN
AL TAN
IBM AND
AL ACOS
AL ASIN
AL ARTN
IBM ATAN
IBM ATN2
AL ARDC
AL ARTN
AL ASIN

*ATAN*AMES*LIBRARY*INVERSE TANGENT OR*ARCTANGENT
*ATAN2*AMES*LIBRARY*INVERSE TANGENT OR*ARCTANGENT
*ATMOSPHERE OF 1959*AMES*LIBRARY*ARDC MODEL
*BACKSPACE*END-OF-FILE MARK ON*MAGNETIC TAP *AMES*LIBRARY*REWIND
*BINARY*TAPE*AMES*LIBRARY*READ
*BOOLEAN OPERATION*AND OR*INTERSECTION*AMES*LIBRARY
*BOOLEAN OPERATION*COMPLEMENT OR*NOT*AMES*LIBRARY
*BOOLEAN OPERATION*OR OR*UNION*AMES*LIBRARY
*BOOLEAN VARIABLE*AMES*LIBRARY*DEFINE FULL LOGICAL WORD
*CLOCK ACCESS PROGRAM*CLOCK*AMES*LIBRARY
*CLOCK*AMES*LIBRARY*CLOCK ACCESS PROGRAM
*COMMON*LOGARITHM OF A NUMBER*ALOG10*AMES*LIBRARY
*COMPLEMENT OR*NOT*AMES*LIBRARY*BOOLEAN OPERATION
*COSH FUNCTION*AMES*LIBRARY*HYPERBOLIC COSINE
*COSINE OF AN ANGLE*AMES*LIBRARY
*COUNT*AMES*LIBRARY*DUMP COUNTING ROUTINE
*CRASH*AMES*LIBRARY*POST MORTEM*DUMP ROUTINE
*CROUT METHOD*AMES*LIBRARY*LINEAR*SIMULTANEOUS EQUATIONS BY
*CURVE FITTING*AMES*LIBRARY*LEAST SQUARES*POLYNOMIAL
*DATA CONVERSION BY ARBITRARY*FORMAT USE*AMES*LIBRARY
*DATE ON OUTPUT*PAGE *AMES*LIBRARY UTILITY PROGRAM FOR*TITLE AN
*DDUMP*AMES*LIBRARY*DYNAMIC*DUMP PROGRAM
*DEFINE FULL LOGICAL WORD*BOOLEAN VARIABLE*AMES*LIBRARY
*DIFFERENTIAL EQUATIONS*AMES*LIBRARY*NUMERICAL SOLUTION OF
*DIFFERENTIAL EQUATIONS*AMES*LIBRARY*NUMERICAL INTEGRATION OF
*DIFFERENTIAL EQUATIONS*AMES*LIBRARY*NUMERICAL INTEGRATION OF
*DIFFERENTIAL EQUATIONS*AMES*LIBRARY*NUMERICAL INTEGRATION OF
*DIFFERENTIATE SET OF DATA POINTS*AMES*LIBRARY*SMOOTH AND
*DOUBLE PRECISION*REAL COEF *AMES*LIBRARY*ROOTS OF A*POLYNOMIA
*DUMP COUNTING ROUTINE*COUNT*AMES*LIBRARY
*DUMP PROGRAM*DDUMP*AMES*LIBRARY*DYNAMIC
*DUMP PROGRAM*PDUMP*AMES*LIBRARY SELECTIVE
*DUMP ROUTINE*AMES*LIBRARY
*DUMP ROUTINE*CRASH*AMES*LIBRARY*POST MORTEM
*DYNAMIC*DUMP PROGRAM*DDUMP*AMES*LIBRARY
*EAI PLOTTER*AMES*LIBRARY PRINT*SCALES ON*PLOTS GENERATED BY
*EAI XY PLOTTER*AMES*LIBRARY*PLOTWS PREPARES TAPE FOR*PLOTS ON
*END-OF-FILE MARK ON*MAGNETIC TAP *AMES*LIBRARY*REWIND*BACKSPAC
*ERROR MESSAGES*AMES*LIBRARY UTILITY PROGRAM FOR
*EXPONENTIAL FUNCTION*AMES*LIBRARY

IBM ATAN
IBM ATN2
AL ARDC
AL NDID
AL BNIN
IBM AND
IBM CMPL
IBM OR
IBM BOOL
AL CLOK
AL CLOK
IBM ALG1
IBM CMPL
AL COSH
IBM COS
AL COUN
AL CRSH
AL CROT
AL LSQP
AL CVRT
AL PAGE
AL DDMP
IBM BOOL
AL ADAM
AL COWL
AL DE6F
AL INT
AL DIF3
AL DPMU
AL COUN
AL DDMP
IBM PDMP
IBM DUMP
AL CRSH
AL DDMP
AL SFAC
AL PLTW
AL NDID
AL EROR
IBM EXP

*FLOATING POINT*OVERFLOW/UNDERFLOW PROGRAM*AMES*LIBRARY	AL	FPT
*FORMAT USE*AMES*LIBRARY*DATA CONVERSION BY ARBITRARY	AL	CVRT
*GAUSS QUAD.*AMES*LIBRARY*NUMERICAL INTEGRATION OF A FUNCTION	AL	GAUS
*HYPERBOLIC COSINE*COSH FUNCTION*AMES*LIBRARY	AL	COSH
*HYPERBOLIC SINE*SINH FUNCTION*AMES*LIBRARY	AL	SINH
*HYPERBOLIC TANGENT*TANH FUNCTION*AMES*LIBRARY	IBM	TANH
*INTEGRATION BY*SIMPSONS RULE*AMES*LIBRARY*NUMERICAL	AL	SIMP
*INTERPOLATION BY*AITKEN METHOD*AMES*LIBRARY*TABLE LOOK-UP AND	AL	TAIN
*INTERSECTION*AMES*LIBRARY*BOOLEAN OPERATION*AND OR	IBM	AND
*INTERVAL HALVING METHOD*AMES*LIBRARY*ROOT OF FUNCTION BY	AL	ITRA
*INTERVAL HALVING METHOD*AMES*LIBRARY*ROOT OF FUNCTION BY	AL	ITR2
*INVERSE COSINE OR*ARCCOSINE*ACOS*AMES*LIBRARY	AL	ACOS
*INVERSE SINE OR*ARCSINE*ASIN*AMES*LIBRARY	AL	ASIN
*INVERSE TANGENT OR*ARCTANGENT*ARTN*AMES*LIBRARY	AL	ARTN
*INVERSE TANGENT OR*ARCTANGENT*ATAN*AMES*LIBRARY	IBM	ATAN
*INVERSE TANGENT OR*ARCTANGENT*ATAN2*AMES*LIBRARY	IBM	ATN2
*LABELS*AMES*LIBRARY*PLOT	AL	ALPP
*LABELS*AMES*LIBRARY*PLOT	AL	PALP
*LEAST SQUARES*POLYNOMIAL*CURVE FITTING*AMES*LIBRARY	AL	LSQP
*LIBRARY COMPUTE*SCALE FACTORS FOR*PLOTS*AMES	AL	SCAL
*LIBRARY DUMMY*PLOT PROGRAM*AMES	AL	PLOT
*LIBRARY DUMMY*PLOT PROGRAM*AMES	AL	PSCA
*LIBRARY EVALUATION OF A*POLYNOMIAL*AMES	AL	POLY
*LIBRARY PRINT*SCALES ON*PLOTS GENERATED BY*EAI PLOTTER*AMES	AL	SFAC
*LIBRARY SELECTIVE*DUMP PROGRAM*PDUMP*AMES	IBM	PDMP
*LIBRARY UTILITY PROGRAM FOR*ERROR MESSAGES*AMES	AL	EROR
*LIBRARY UTILITY PROGRAM*AMES	AL	EDFL
*LIBRARY UTILITY PROGRAM FOR*TITLE AND*DATE ON OUTPUT*PAGE *AME	AL	PAGE
*LIBRARY UTILITY PROGRAM FOR POSITIONING*TAPES*LOCATE*AMES	AL	LOCA
*LIBRARY UTILITY PROGRAM TO*REWIND AND UNLOAD MAG*TAPE*AMES	AL	REWN
*LIBRARY*ARDC MODEL*ATMOSPHERE OF 1959*AMES	AL	ARDC
*LIBRARY*BOOLEAN OPERATION*COMPLEMENT OR*NOT*AMES	IBM	CMPL
*LIBRARY*BOOLEAN OPERATION*AND OR*INTERSECTION*AMES	IBM	AND
*LIBRARY*BOOLEAN OPERATION*OR OR*UNION*AMES	IBM	OR
*LIBRARY*CLOCK ACCESS PROGRAM*CLOCK*AMES	AL	CLOK
*LIBRARY*COMMON*LOGARITHM OF A NUMBER*ALOG10*AMES	IBM	ALG1
*LIBRARY*COSINE OF AN ANGLE*AMES	IBM	COS
*LIBRARY*DATA CONVERSION BY ARBITRARY*FORMAT USE*AMES	AL	CVRT
*LIBRARY*DEFINE FULL LOGICAL WORD*BOOLEAN VARIABLE*AMES	IBM	BOOL
*LIBRARY*DUMP COUNTING ROUTINE*COUNT*AMES	AL	COUN

*LIBRARY*DUMP ROUTINE*AMES
 *LIBRARY*DYNAMIC*DUMP PROGRAM*DDUMP*AMES
 *LIBRARY*EXPONENTIAL FUNCTION*AMES
 *LIBRARY*FLOATING POINT*OVERFLOW/UNDERFLOW PROGRAM*AMES
 *LIBRARY*HYPERBOLIC COSINE*COSH FUNCTION*AMES
 *LIBRARY*HYPERBOLIC SINE*SINH FUNCTION*AMES
 *LIBRARY*HYPERBOLIC TANGENT*TANH FUNCTION*AMES
 *LIBRARY*INVERSE COSINE OR*ARCCOSINE*ACOS*AMES
 *LIBRARY*INVERSE SINE OR*ARCSINE*ASIN*AMES
 *LIBRARY*INVERSE TANGENT OR*ARCTANGENT*ARTN*AMES
 *LIBRARY*INVERSE TANGENT OR*ARCTANGENT*ATAN2*AMES
 *LIBRARY*INVERSE TANGENT OR*ARCTANGENT*ATAN*AMES
 *LIBRARY*LEAST SQUARES*POLYNOMIAL*CURVE FITTING*AMES
 *LIBRARY*LINEAR*SIMULTANEOUS EQUATIONS BY*CROUT METHOD*AMES
 *LIBRARY*MATRIX INVERSION AND*SIMULTANEOUS LINEAR EQUATION *AME
 *LIBRARY*MATRIX INVERSION AND*SIMULTANEOUS EQUATIONS SOLUT *AME
 *LIBRARY*MATRIX INVERSION AND*SIMULTANEOUS EQN SOLUTION*AMES
 *LIBRARY*NATURAL*LOGARITHM OF A NUMBER*ALOG*AMES
 *LIBRARY*NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS*AMES
 *LIBRARY*NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS*AMES
 *LIBRARY*NUMERICAL INTEGRATION OF A FUNCTION*GAUSS QUAD.*AMES
 *LIBRARY*NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS*AMES
 *LIBRARY*NUMERICAL SOLUTION OF*DIFFERENTIAL EQUATIONS*AMES
 *LIBRARY*NUMERICAL*INTEGRATION BY*SIMPSON'S RULE*AMES
 *LIBRARY*PARITY OF AN INTEGER*XPAR*AMES
 *LIBRARY*PLOT*LABELS*AMES
 *LIBRARY*PLOT*LABELS*AMES
 *LIBRARY*PLOTWS PREPARES TAPE FOR*PLOTS ON*EAI XY PLOTTER*AMES
 *LIBRARY*POST MORTEM*DUMP ROUTINE*CRASH*AMES
 *LIBRARY*RANDOM NUMBER GENERATOR*AMES
 *LIBRARY*RANDOM NUMBER GENERATOR*AMES
 *LIBRARY*RANDOM NUMBER GENERATOR*AMES
 *LIBRARY*READ*BINARY*TAPE*AMES
 *LIBRARY*REWIND*BACKSPACE*END-OF-FILE MARK ON*MAGNETIC TAP *AME
 *LIBRARY*ROOT OF AN ARBITRARY FUNCTION*NEWTON-RAPHSON*AMES
 *LIBRARY*ROOT OF FUNCTION BY*INTERVAL HALVING METHOD*AMES
 *LIBRARY*ROOT OF FUNCTION BY*INTERVAL HALVING METHOD*AMES
 *LIBRARY*ROOTS OF A*POLYNOMIAL*DOUBLE PRECISION*REAL COEF *AME
 *LIBRARY*ROOTS OF A*POLYNOMIAL*AMES
 *LIBRARY*ROOTS OF*POLYNOMIALS WITH REAL COEFFICIENTS*AMES

IBM DUMP
 AL DDMP
 IBM EXP
 AL FPT
 AL COSH
 AL SINH
 IBM TANH
 AL ACOS
 AL ASIN
 AL ARTN
 IBM ATN2
 IBM ATAN
 AL LSQP
 AL CROT
 AL MINV
 AL INVT
 AL MTNV
 IBM ALOG
 AL COWL
 AL DE6F
 AL GAUS
 AL INT
 AL ADAM
 AL SIMP
 AL XPAR
 AL ALPP
 AL PALP
 AL PLTW
 AL CRSH
 AL BARN
 AL RNDM
 AL RDM
 AL BNIN
 AL NDID
 AL ROOT
 AL ITRA
 AL ITR2
 AL DPMU
 AL ROP
 ML HPRS

*LIBRARY*SIMULTANEOUS*LINEAR EQUATIONS*AMES
 *LIBRARY*SINE OF AN ANGLE*AMES
 *LIBRARY*SMOOTH AND*DIFFERENTIATE SET OF DATA POINTS*AMES
 *LIBRARY*SQUARE ROOT FUNCTION*AMES
 *LIBRARY*TABLE LOOK-UP AND*INTERPOLATION BY*AITKEN METHOD*AMES
 *LIBRARY*TANGENT FUNCTION*AMES
 *LINEAR EQUATIONS*AMES*LIBRARY*SIMULTANEOUS
 *LINEAR*SIMULTANEOUS EQUATIONS BY*CROUT METHOD*AMES*LIBRARY
 *LOCATE*AMES*LIBRARY UTILITY PROGRAM FOR POSITIONING*TAPES
 *LOGARITHM OF A NUMBER*ALOG10*AMES*LIBRARY*COMMON
 *LOGARITHM OF A NUMBER*ALOG*AMES*LIBRARY*NATURAL
 *MAGNETIC TAP *AMES*LIBRARY*REWIND*BACKSPACE*END-OF-FILE MARK 0
 *MATRIX INVERSION AND*SIMULTANEOUS EQN SOLUTION*AMES*LIBRARY
 *MATRIX INVERSION AND*SIMULTANEOUS LINEAR EQUATION *AMES*LIBRAR
 *MATRIX INVERSION AND*SIMULTANEOUS EQUATIONS SOLUT *AMES*LIBRAR
 *NATURAL*LOGARITHM OF A NUMBER*ALOG*AMES*LIBRARY
 *NEWTON-RAPHSON*AMES*LIBRARY*ROOT OF AN ARBITRARY FUNCTION
 *NOT*AMES*LIBRARY*BOOLEAN OPERATION*COMPLEMENT OR
 *NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS*AMES*LIBRARY
 *NUMERICAL INTEGRATION OF A FUNCTION*GAUSS QUAD.*AMES*LIBRARY
 *NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS*AMES*LIBRARY
 *NUMERICAL INTEGRATION OF*DIFFERENTIAL EQUATIONS*AMES*LIBRARY
 *NUMERICAL SOLUTION OF*DIFFERENTIAL EQUATIONS*AMES*LIBRARY
 *NUMERICAL*INTEGRATION BY*SIMPSON'S RULE*AMES*LIBRARY
 *OR OR*UNION*AMES*LIBRARY*BOOLEAN OPERATION
 *OVERFLOW/UNDERFLOW PROGRAM*AMES*LIBRARY*FLOATING POINT
 *PAGE *AMES*LIBRARY UTILITY PROGRAM FOR*TITLE AND*DATE ON OUTPU
 *PARITY OF AN INTEGER*XPAR*AMES*LIBRARY
 *PDUMP*AMES*LIBRARY SELECTIVE*DUMP PROGRAM
 *PLOT PROGRAM*AMES*LIBRARY DUMMY
 *PLOT PROGRAM*AMES*LIBRARY DUMMY
 *PLOT*LABELS*AMES*LIBRARY
 *PLOT*LABELS*AMES*LIBRARY
 *PLOTS GENERATED BY*EAI PLOTTER*AMES*LIBRARY PRINT*SCALES ON
 *PLOTS ON*EAI XY PLOTTER*AMES*LIBRARY*PLOTWS PREPARES TAPE FOR
 *PLOTS*AMES*LIBRARY COMPUTE*SCALE FACTORS FOR
 *PLOTWS PREPARES TAPE FOR*PLOTS ON*EAI XY PLOTTER*AMES*LIBRARY
 *POLYNOMIAL*AMES*LIBRARY*ROOTS OF A
 *POLYNOMIAL*AMES*LIBRARY EVALUATION OF A
 *POLYNOMIAL*CURVE FITTING*AMES*LIBRARY*LEAST SQUARES

AL LSQS
 IBM SIN
 AL DIF3
 IBM SORT
 AL TAIN
 AL TAN
 AL LSQS
 AL CROT
 AL LOCA
 IBM ALG1
 IBM ALOG
 AL NDID
 AL MTNV
 AL MINV
 AL INVT
 IBM ALOG
 AL ROOT
 IBM CMPL
 AL COWL
 AL GAUS
 AL DE6F
 AL INT
 AL ADAM
 AL SIMP
 IBM OR
 AL FPT
 AL PAGE
 AL XPAR
 IBM PDMP
 AL PSCA
 AL PLOT
 AL ALPP
 AL PALP
 AL SFAC
 AL PLTW
 AL SCAL
 AL PLTW
 AL ROP
 AL POLY
 AL LSQP

*POLYNOMIAL*DOUBLE PRECISSION*REAL COEF *AMES*LIBRARY*ROOTS OF	AL	DPMU
*POLYNOMIALS WITH REAL COEFFICIENTS*AMES*LIBRARY*ROOTS OF	ML	HPRS
*POST MORTEM*DUMP ROUTINE*CRASH*AMES*LIBRARY	AL	CRSH
*RANDOM NUMBER GENERATOR*AMES*LIBRARY	AL	BARN
*RANDOM NUMBER GENERATOR*AMES*LIBRARY	AL	RNDM
*RANDOM NUMBER GENERATOR*AMES*LIBRARY	AL	RDM
*READ*BINARY*TAPE*AMES*LIBRARY	AL	BNIN
*REAL COEF *AMES*LIBRARY*ROOTS OF A*POLYNOMIAL*DOUBLE PRECISSIO	AL	DPMU
*REWIND AND UNLOAD MAG*TAPE*AMES*LIBRARY UTILITY PROGRAM TO	AL	REWN
*REWIND*BACKSPACE*END-OF-FILE MARK ON*MAGNETIC TAP *AMES*LIBRAR	AL	NDID
*ROOT OF AN ARBITRARY FUNCTION*NEWTON-RAPHSON*AMES*LIBRARY	AL	ROOT
*ROOT OF FUNCTION BY*INTERVAL HALVING METHOD*AMES*LIBRARY	AL	ITR2
*ROOT OF FUNCTION BY*INTERVAL HALVING METHOD*AMES*LIBRARY	AL	ITRA
*ROOTS OF A*POLYNOMIAL*DOUBLE PRECISSION*REAL COEF *AMES*LIBRAR	AL	DPMU
*ROOTS OF A*POLYNOMIAL*AMES*LIBRARY	AL	ROP
*ROOTS OF*POLYNOMIALS WITH REAL COEFFICIENTS*AMES*LIBRARY	ML	HPRS
*SCALE FACTORS FOR*PLOTS*AMES*LIBRARY COMPUTE	AL	SCAL
*SCALES ON*PLOTS GENERATED BY*EAI PLOTTER*AMES*LIBRARY PRINT	AL	SFAC
*SIMPSONS RULE*AMES*LIBRARY*NUMERICAL*INTEGRATION BY	AL	SIMP
*SIMULTANEOUS EQN SOLUTION*AMES*LIBRARY*MATRIX INVERSION AND	AL	MTNV
*SIMULTANEOUS EQUATIONS BY*CROUT METHOD*AMES*LIBRARY*LINEAR	AL	CROT
*SIMULTANEOUS EQUATIONS SOLUT *AMES*LIBRARY*MATRIX INVERSION AN	AL	INVT
*SIMULTANEOUS LINEAR EQUATION *AMES*LIBRARY*MATRIX INVERSION AN	AL	MINV
*SIMULTANEOUS*LINEAR EQUATIONS*AMES*LIBRARY	AL	LSQS
*SINE OF AN ANGLE*AMES*LIBRARY	IBM	SIN
*SINH FUNCTION*AMES*LIBRARY*HYPERBOLIC SINE	AL	SINH
*SMOOTH AND*DIFFERENTIATE SET OF DATA POINTS*AMES*LIBRARY	AL	DIF3
*SQUARE ROOT FUNCTION*AMES*LIBRARY	IBM	SQRT
*TABLE LOOK-UP AND*INTERPOLATION BY*AITKEN METHOD*AMES*LIBRARY	AL	TAIN
*TANGENT FUNCTION*AMES*LIBRARY	AL	TAN
*TANH FUNCTION*AMES*LIBRARY*HYPERBOLIC TANGENT	IBM	TANH
*TAPE*AMES*LIBRARY UTILITY PROGRAM TO*REWIND AND UNLOAD MAG	AL	REWN
*TAPE*AMES*LIBRARY*READ*BINARY	AL	BNIN
*TAPES*LOCATE*AMES*LIBRARY UTILITY PROGRAM FOR POSITIONING	AL	LOCA
*TITLE AND*DATE ON OUTPUT*PAGE *AMES*LIBRARY UTILITY PROGRAM FO	AL	PAGE
*UNION*AMES*LIBRARY*BOOLEAN OPERATION*OR OR	IBM	OR
*XPAR*AMES*LIBRARY*PARITY OF AN INTEGER	AL	XPAR

- B1 ELEMENTARY FUNCTIONS, TRIGONOMETRIC
 AL ACOS AMES LIBRARY-INVERSE COSINE FUNCTION
 AL ARTN AMES LIBRARY-INVERSE TANGENT FUNCTION
 AL ASIN AMES LIBRARY-INVERSE SINE FUNCTION
 AL TAN AMES LIBRARY TANGENT FUNCTION
 IBM ATAN AMES LIBRARY-INVERSE TANGENT FUNCTION
 IBM ATN2 AMES LIBRARY-INVERSE TANGENT FUNCTION ATAN2
 IBM COS AMES LIBRARY COSINE OF AN ANGLE
 IBM SIN AMES LIBRARY SINE OF AN ANGLE
- B2 ELEMENTARY FUNCTIONS, HYPERBOLIC
 AL COSH AMES LIBRARY HYPERBOLIC COSINE FUNCTION COSH
 AL SINH AMES LIBRARY HYPERBOLIC SINE FUNCTION SINH
 IBM TANH AMES LIBRARY HYPERBOLIC TANGENT FUNCTION
- B3 ELEMENTARY FUNCTIONS, EXPONENTIAL AND LOGARITHMIC
 IBM ALG1 AMES LIBRARY ALOG10-COMPUTE COMMON LOGARITHMS
 IBM ALOG AMES LIBRARY-COMPUTE NATURAL LOGARITHM
 IBM EXP AMES LIBRARY EXPONENTIAL FUNCTION
- B4 ELEMENTARY FUNCTIONS, ROOTS AND POWERS
 IBM SQRT AMES LIBRARY SQUARE ROOT FUNCTION
- C1 POLYNOMIALS + SPECIAL FUNCTIONS, EVALUATION OF POLYNOMIALS
 AL POLY AMES LIBRARY EVALUATION OF A POLYNOMIAL
- C2 POLYNOMIALS + SPECIAL FUNCTIONS, ROOTS OF POLYNOMIALS
 AL DPMU AMES LIBRARY-ROOTS OF A POLYNOMIAL IN DOUBLE PRECISION
 AL ROP AMES LIBRARY-ROOTS OF A POLYNOMIAL
 ML HPRS AMES LIBRARY-ROOTS OF POLYNOMIAL WITH REAL COEFFICIENTS
- C3 POLYNOMIALS + SPECIAL FUNCTIONS, EVALUATION OF SPECIAL FUNCTIONS
 AL ARDC AMES LIBRARY-ARDC MODEL ATMOSPHERE OF 1959
- C5 POLYNOMIALS + SPECIAL FUNC. SIMULTANEOUS TRANSCENDENTAL EQUATIONS
 AL ITRA AMES LIBRARY-ROOT OF FUNCTION BY INTERVAL HALVING METHOD
 AL ITR2 AMES LIBRARY-ROOT OF FUNCTION BY INTERVAL HALVING METHOD
 AL ROOT AMES LIBRARY ROOT OF AN ARBITRARY FUNCTION
- D1 OPER ON FUNC + SOLUTIONS OF DIFFERENTIAL EQUA, NUMERICAL INTEGRATION

D1	OPER ON FUNC + SOLUTIONS OF DIFFERENTIAL EQUA., NUMERICAL INTEGRATION	
	AL GAUS	AMES LIBRARY NUMERICAL INTEGRATION PROGRAM GAUSS
	AL SIMP	AMES LIBRARY-NUMERICAL INTEGRATION BY SIMPSON'S RULE
D2	NUMERICAL SOLUTIONS OF ORDINARY DIFFERENTIAL EQUATIONS	
	AL ADAM	AMES LIBRARY NUMERICAL SLN OF DIFFERENTIAL EQUATIONS
	AL COWL	AMES LIBRARY-NUMERICAL INTEGRATION OF DIFFERENTIAL EQUATI
	AL DE6F	AMES LIBRARY NUMERICAL INTEGRATION OF DIFFERENTIAL EQNS
	AL INT	AMES LIBRARY NUMERICAL INTEGRATION OF DIFFERENTIAL EQNS
D4	OPER OF FUNC + SOLUTIONS OF DIFFERENTIAL EQUA. NUMERICAL DIFFEREN.	
	AL DIF3	AMES LIBRARY SMOOTH AND DIFFERENTIATE SET OF DATA POINTS
E1	INTERPOLATION AND APPROXIMATIONS, TABLE LOOK-UP AND INTERPOLATION	
	AL TAIN	AMES LIBRARY-TABLE LOOK-UP AND INTERPOLATION, BINARY SEARC
E2	INTERPOLATION AND APPROXIMATIONS, CURVE FITTING	
	AL LSQP	AMES LIBRARY LEAST SQUARES POLYNOMIAL CURVE FITTING
E3	INTERPOLATION AND APPROXIMATIONS, SMOOTHING	
	AL DIF3	AMES LIBRARY SMOOTH AND DIFFERENTIATE SET OF DATA POINTS
F1	OPER ON MATRICES, VECTORS + SIMUL LINEAR EQUA. MATRIX OPERATIONS	
	AL INVT	AMES LIBRARY-MATRIX INVERSION AND SIMULTANEOUS EQN SOLUTI
	AL MINV	AMES LIBRARY-MATRIX INVERSION AND SIMULTANEOUS LINEAR EQN
	AL MTNV	AMES LIBRARY MATRIX INVERSION AND SIMULTANEOUS EQUATIONS
F4	OPER ON MATRICES, VECTORS + SIMUL LINEAR EQUA., SIMUL LINEAR EQUA.	
	AL CROT	AMES LIBRARY-LINEAR SIMULTANEOUS EQNS BY CROUT METHOD
	AL INVT	AMES LIBRARY-MATRIX INVERSION AND SIMULTANEOUS EQN SOLUTI
	AL LSQS	AMES LIBRARY-SIMULTANEOUS LINEAR EQUATIONS
	AL MINV	AMES LIBRARY-MATRIX INVERSION AND SIMULTANEOUS LINEAR EQN
	AL MTNV	AMES LIBRARY MATRIX INVERSION AND SIMULTANEOUS EQUATIONS
I4	INPUT, BCD HOLLERITH	
	AL BNIN	AMES LIBRARY- READ BINARY TAPE
J0	OUTPUT	
	AL LOCA	AMES LIBRARY TAPE POSITIONING PROGRAM LOCATE
	AL NDID	AMES LIBRARY-REWIND, BACKSPACE AND EOF ON MAGNETIC TAPE

J4 OUTPUT, BCD HOLLERITH
AL PAGE AMES LIBRARY UTILITY PROGRAM FOR TITLE AND DATE ON DATA
AL PLTW AMES LIBRARY PLOTWS PREPARES PLOT TAPE FOR EAI PLOTTER

J5 OUTPUT, PLOTTING
AL ALPP AMES LIBRARY -PLOT LABELLING ROUTINE
AL PALP AMES LIBRARY-PLOT LABELLING ROUTINE
AL PLOT AMES LIBRARY DUMMY PLOT PROGRAM
AL PSCA AMES LIBRARY DUMMY PLOT PROGRAM
AL SCAL AMES LIBRARY-COMPUTE SCALE FACTORS FOR PLOTS
AL SFAC AMES LIBRARY-PRINT SCALES ON PLOTS MADE ON EAI DATAPLOTTE

K2 INTERNAL INFORMATION TRANSFER, RELOCATION
AL CVRT AMES LIBRARY-DATA CONVERSION BY ARBITRARY FORMAT USE

M2 DATA HANDLING, CONVERSION AND/OR SCALING
AL CVRT AMES LIBRARY-DATA CONVERSION BY ARBITRARY FORMAT USE

M4 DATA HANDLING, CHARACTER MANIPULATION
AL CVRT AMES LIBRARY-DATA CONVERSION BY ARBITRARY FORMAT USE

N2 DEBUGGING, DUMPING
AL COUN AMES LIBRARY-DUMP COUNTING ROUTINE
AL CRSH AMES LIBRARY-POST MORTEM DUMPING PROGRAM CRASH
AL DDMP AMES LIBRARY DYNAMIC DUMPING PROGRAM DDUMP
IBM DUMP AMES LIBRARY DUMP ROUTINE
IBM PDMP AMES LIBRARY PDUMP PROGRAM

Q3 REWIND, TAPE MARK, LOAD CARDS, LOAD TAPE, ETC. PROGRAMS
AL REWN AMES LIBRARY UTILITY PROGRAM TO REWIND AND UNLOAD TAPE

V1 GEN PURPOSE UTILITY SUBROUTINES, RANDOM NUMBER GENERATORS
AL BARN AMES LIBRARY- RANDOM NUMBER GENERATOR
AL RDM AMES LIBRARY-RANDOM NUMBER GENERATOR
AL RNDM AMES LIBRARY-RANDOM NUMBER GENERATOR

Z0 ALL OTHERS
AL CLOK AMES LIBRARY CLOCK ACCESS PROGRAM
AL EDFL AMES LIBRARY UTILITY PROGRAM
AL EROR AMES LIBRARY UTILITY PROGRAM FOR ERROR MESSAGES ERROR

ZO

ALL OTHERS

AL	FPT	AMES	LIBRARY FLOATING POINT OVERFLOW PROGRAM
AL	XPAR	AMES	LIBRARY-DETERMINATION OF PARITY OF AN INTEGER
IBM	AND	AMES	LIBRARY-BOOLEAN OPERATION AND OR INTERSECTION
IBM	BOOL	AMES	LIBRARY-DEFINE FULL LOGICAL WORD (BOOLEAN)
IBM	CMPL	AMES	LIBRARY-COMPL-BOOLEAN OPERATION COMPLEMENT OR NOT
IBM	OR	AMES	LIBRARY-BOOLEAN OPERATION OR OR UNION

SUMMARIES

Identification

AL ACOS--ASIN, Inverse Sine and Cosine Functions
FORTRAN IV, MAP-coded
Ames Modification of SHARE Library Routine NA135.3

(See writeup for AL ASIN-ACOS).

Identification

AL ADAM, Numerical Integration of First Order Differential Equations by the
Adams Method

FORTTRAN IV

Written by J. A. Jeske

Purpose

This subroutine is used to obtain the numerical solution of the system of N ordinary differential equations

$$y_i' = f_i(x, y_i) \quad (i = 1, 2, \dots, N),$$

where the prime indicates differentiation with respect to the independent variable x. The initial conditions are:

$$y_1(x_0) = (y_1)_0$$

$$y_2(x_0) = (y_2)_0$$

$$\vdots$$

$$y_n(x_0) = (y_n)_0$$

Both forward and backward starting procedures are provided.

Usage

This subroutine is entered via

CALL ADAMS (X, T, DERIV, H, M, N),

where

- | | |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| X | is the independent variable |
| T | is a two dimensional array that contains values of y_i and y_i' (see table below), T must be dimensioned (9,N) in the calling program |
| DERIV(X,T) | is a user supplied subroutine that computes the first derivatives y_1', y_2', \dots, y_n' and stores them in T(2,1), T(2,2), ..., T(2,N) the name DERIV (or whatever name the user chooses) must appear in an EXTERNAL specification statement in the calling program. |
| H | is the interval in X over which integration is to take place. |
| M | is a special code where
M = -1 backward starter
M = 0 forward starter
M > 1 indicates that integration is to be performed from X to X + MH. |
| N | is the number of differential equations in the system. |

Note that for M = -1 or M = 0, only a setup function is performed. The subroutine must be called using one of the starting procedures before integration can be performed. Each call where integration is specified (M > 1) can be carried out without reinitiating a starting procedure. If, however, the interval size H is to be changed, a new start must be made.

The array T is loaded as follows:

$$\begin{array}{l}
 \left. \begin{array}{l}
 T(1,1) = Y_1 \\
 T(1,2) = Y_2 \\
 \vdots \\
 T(1,N) = Y_n
 \end{array} \right\} \text{The user places initial values here prior to a} \\
 \text{starter entry} \\
 \\
 \left. \begin{array}{l}
 T(2,1) = Y'_1 \\
 T(2,2) = Y'_2 \\
 \vdots \\
 T(2,N) = Y'_n
 \end{array} \right\} \text{The user's subroutine DERIV places first derivatives here.}
 \end{array}$$

If it is desired to examine the backward differences ∇f_I , $\nabla^2 f_I$, ..., $\nabla^5 f_I$ after any entry, they may be found in $T(3,I)$, $T(4,I)$, ..., $T(7,I)$, where $I = 1, 2, \dots, N$.

Method

A fourth-order Adams first sum predictor-corrector method is used for continued integration, and an iterated fourth order Adams first sum method is used for the starters (see reference).

The predictor is given by the equations

$$\begin{aligned}
 y_{m+1}^* &= A_m + h \sum_{k=0}^4 \gamma_{-1,k+1} \nabla^k f_m \\
 f_{m+1}^* &= f(x_{m+1}, y_{m+1}^*)
 \end{aligned}$$

and the corrector by

$$\begin{aligned}
 y_{m+1} &= y_{m+1}^* + h \gamma_{-1,5} \nabla^5 f_{m+1}^* \\
 f_{m+1} &= f(x_{m+1}, y_{m+1}) \\
 A_{m+1} &= A_m + h f_{m+1}
 \end{aligned}$$

where $\gamma_{-1,j}$ are constants and A_m is the first sum. Note that in the above equations m corresponds to x_m at the m th integration step. Also, the equations are written in vector form, that is $y_m = (y_{m,1}, y_{m,2}, \dots, y_{m,n})$.

The backward starter iterates the following eight times:

$$\begin{aligned}
 p &= 1(1)4 \\
 y_{-p} &= A_0 + h \sum_{m=0}^4 \gamma_{4,p,m} f_{-m} \\
 f_{-p} &= f(x_{-p}, y_{-p}) \\
 A_0 &= y_0 - h \sum_{m=0}^4 \gamma_{4,0,m} f_{-m}
 \end{aligned}$$

where the $\gamma_{4,i,j}$ are constants.

Initial values are given by

$$A_0 = y_0 + \frac{1}{2}hf_0.$$

The ordinates $y_0, y_{-1}, \dots, y_{-4}$ are then converted to backward differences $\nabla f_0, \nabla^2 f_0, \dots, \nabla^4 f_0$.

The equations for the backward starter are transformed to those at the forward starter by means of the transformation

$$\begin{aligned} h &\rightarrow -h \\ y_{-m} &\rightarrow y_m \\ f_{-m} &\rightarrow f_m \end{aligned}$$

The resulting ordinates y_0, y_1, \dots, y_4 are then converted to backward differences $\nabla f_4, \nabla^2 f_4, \dots, \nabla^4 f_4$ at $x = x_0 + 4h$. Shifting these differences to x_0 is accomplished by assuming $\nabla^5 f_4 = 0$. The final result is the backward differences $\nabla f_0, \nabla^2 f_0, \dots, \nabla^4 f_0$.

A more complete discussion of the method and the values of all the constants may be found in the reference.

Reference

Mersman, William A: Self-starting Multi-step Methods for the Numerical Integration of Ordinary Differential Equations.
NASA TN D-2936, 1965

Identification

AL ALPP, Prints Alphanumeric and Special Characters on EAI Dataplotter
Plots Using a 48-Character Symbol Printer
FORTRAN IV, MAP-coded
Written by Melba Perniciaro

Purpose

This subroutine is used to print any of 48 alphanumeric and special characters (including blank) at a desired location in either the X or Y direction. Its primary use is the titling and annotation of plots prepared with the FORTRAN IV routines AL PLOTWS or AL MPLT, using an EAI Dataplotter equipped with a 48-character symbol printer.

Usage

The calling sequence for AL ALPP is:

CALL ALPP (ORIGX, ORIGY, ARRAY, NUM, NTAPE, NXORY)

The symbols in the CALL statement are defined as follows:

ORIGX } ORIGY }	The horizontal and vertical coordinates, respectively, (measured in signed inches from the center of the page of plotting paper) of the first character to be printed.
ARRAY	An array containing the alphanumeric characters (including blanks) to be printed, 6 characters/word, or a Hollerith string of the form nH..., where n=NUM.
NUM	The number of characters (including blanks) to be printed.
NTAPE	The logical number of the plot tape.
NXORY	A code word controlling the direction the characters are printed: If NXORY = +1, the characters are printed in the X direction. If NXORY = -1, the characters are printed in the Y direction.

Restrictions

The argument "ARRAY" in the calling sequence must be one of the following:

1. A Hollerith string of the form nH..., where n is the number of characters (including blanks) to be printed.

Example: CALL ALPP (ORIGX, ORIGY,
23HTHIS IS A SAMPLE TITLE.,23,NTAPE,NXORY)

2. An array containing alphanumeric information in BCD form, 6 characters/word.
The array can be filled from cards using the "A" format or with a DATA statement.

In this routine, the Hollerith character "blank" must be used for spacing. Thus, the special character \diamond (which corresponds to "blank" in the FORTRAN IV library routine AL PLOTWS) is not available in AL ALPP.

Although the plotting paper is approximately 30 inches square, the actual area available for plotting is quite different. In the X direction, we must allow an inch on either side of the paper to keep the pen and printer arms from colliding. In the Y direction, we must allow 1-3/4" on either side for the width of the printing attachment. Thus, any plotting information, and thus all characters to be printed by AL ALPP, must be confined to an area of 28 inches by 26½ inches.

Discussion

The 48 characters available are:

Hollerith character

Symbol

0 - 9

0 - 9

A - Z

A - Z

.

.

,

,

(

(

)

)

=

=

+

+

- (minus)

- (minus)

*

*

/

/

- (dash)

□

\$

○

blank

spaces the printer 1/10"

The first character printed is more or less centered about the point (ORIGX, ORIGY), followed by the other characters, printed in the direction specified. Corresponding points on adjacent characters are 1/10" apart, and spaces between words (i.e., the character "blank") are also 1/10".

For maximum efficiency this routine is written to use the "select and print" mode of the plotter. In this mode there is virtually no delay for changing the character to be printed since this is done as the plotter moves to the point where the new symbol is to be written.

Examples (See attached sheets)

- I. The characters along the top and right sides of the page illustrate the actual appearance of all the characters available. For the top, a DATA statement was used to fill an array ALPHA (dimension 8) with the Hollerith information desired.

```
DATA(ALPHA(IO),IO=1,8)/48H0123...)-$/
CALL ALPP (0.0, 5.75, ALPHA, 48, 7, +1)
```

For the right side, a string of Hollerith characters was used in the CALL statement:

```
CALL ALPP (6.5, 1.5, 48H0123...)-$, 48, 7, -1)
```

For the graph at the bottom of the page, AL PLOT was used for the curve and axes; AL SFAC produced the scale factors on the axes; and AL ALPP labelled the plot. In this case, the characters to be printed by AL ALPP were read into arrays OMS (dimension 4), CYC (dimension 5), and CAP (dimension 2) from cards, using the "A" format. The calling sequences for AL ALPP were:

```
CALL ALPP (-0.5, 1.5, OMS, 23, 7, -1)
CALL ALPP ( 0.5, 0.25, CYC, 29, 7, +1)
CALL ALPP (4.0, 3.5, CAP, 11, 7, +1)
```

- II. The second page of examples shows the labels to be used on a set of 18 plots.

```

        DIMENSION STALOC(2,20),IDATE(2,20),ITIME(20),ARRAY(5),ORIG(2,20),
1  WORK(5),IWORK(5),FMT1(4),FMT2(1)
        EQUIVALENCE (WORK(1),IWORK(1))
        DATA(FMT1(IO),IO=1,4)/24H(2A6,2H, ,A6,A2,2H, ,I4)/,FMT2(1)/
1  5H(5A6)/
        :
        KOUNT=0
        DO 101 I=1,NOSTA
        WORK(1)=STALOC(1,I)
        WORK(2)=STALOC(2,I)
        DO 101 J=1,NODATE
        IWORK(3)=IDATE(1,J)
        IWORK(4)=IDATE(2,J)
        DO 101 K=1,NOTIME
        IWORK(5)=ITIME(K)
        CALL CVRT (WORK,5,FMT1,ARRAY,5,FMT2)
        KOUNT=KOUNT+1
        CALL ALPP (ORIG(KOUNT,1),ORIG(KOUNT,2),ARRAY,28,7,+1)
101  CONTINUE

```

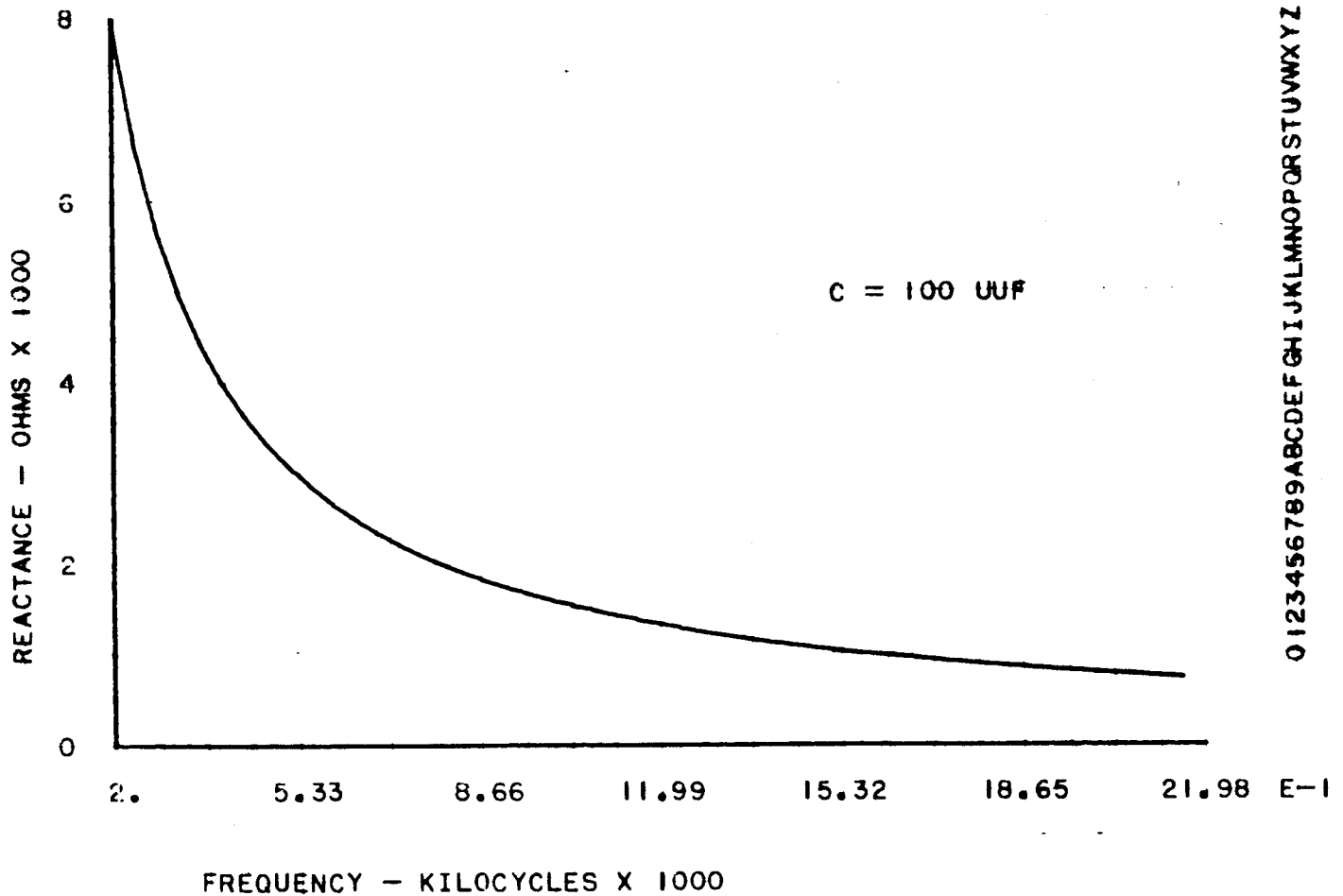
Note that the numerical information in the labels was generated within the program.
It was converted to the proper form by using the FORTRAN IV subroutine AL CVRT.

References

1. FORTRAN IV library routines AL PLTW(PLOTWS) and AL PLOT.
2. FORTRAN IV routine AL MPLT.
3. FORTRAN IV routine AL SFAC.
4. FORTRAN IV routine AL CVRT.

0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z = + * / . , () E O

0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z = + * / . , () E O



JOHANNESBURG, 12/01/64, 930

JOHANNESBURG, 12/01/64, 1530

JOHANNESBURG, 04/16/66, 930

JOHANNESBURG, 04/16/66, 1530

JOHANNESBURG, 08/31/68, 930

JOHANNESBURG, 08/31/68, 1530

PUERTO RICO , 12/01/64, 930

PUERTO RICO , 12/01/64, 1530

PUERTO RICO , 04/16/66, 930

PUERTO RICO , 04/16/66, 1530

PUERTO RICO , 08/31/68, 930

PUERTO RICO , 08/31/68, 1530

GRAND BAHAMA, 12/01/64, 930

GRAND BAHAMA, 12/01/64, 1530

GRAND BAHAMA, 04/16/66, 930

GRAND BAHAMA, 04/16/66, 1530

GRAND BAHAMA, 08/31/68, 930

GRAND BAHAMA, 08/31/68, 1530

Identification

AL ARDC (ARDC1) Altitude Function Subroutine
 FORTRAN IV, MAP-coded
 Ames Modification of G.E. Program Library No. 704-082,
 Altitude Function Subroutine

Purpose

This subroutine is used to calculate the functions of geometric altitude given by the ARDC Model Atmosphere of 1959.

Usage

This subroutine is entered by use of the statement

```
CALL ARDC1(ALT,ANS)
```

where

ALT is the altitude above the surface of the earth, in meters.
 (ALT \leq 10^6 meters)

ANS is an array of dimension 12 that contains the results
 upon return from the subroutine

The arrangement, in ANS, of the computed quantities is given next.

<u>Location</u>	<u>Quantity</u>	<u>Units</u>
ANS(1)	pressure/pressure at sea level	
ANS(2)	acceleration of gravity	M/sec ²
ANS(3)	velocity of sound	M/sec
ANS(4)	kinematic viscosity	M ² /sec
ANS(5)	viscosity/viscosity at sea level	
ANS(6)	viscosity	KG(mass)/M sec
ANS(7)	density/density at sea level	
ANS(8)	density	KG(force)sec ² /M ⁴
ANS(9)	pressure at sea level minus pressure at given altitude	KG(force)/M ²
ANS(10)	pressure	KG(force)/M ²
ANS(11)	real kinetic temperature	deg Kelvin
ANS(12)	molecular temperature	deg Kelvin

Usage (continued)

The definitions

KG(force)	force in kilograms
KG(mass)	mass in kilograms
M	meters
sec	seconds

are employed in the foregoing. If the density is required in the units $\text{KG}(\text{force})/\text{M}^3$, it may be obtained directly from the product $\text{ANS}(8) * \text{ANS}(2)$.

Restrictions

The velocity of sound, kinematic viscosity, viscosity/viscosity at sea-level, and the viscosity are not determined, and cells $\text{ANS}(3-6)$ are set to zero, for ALT above 91,293 meters. The input altitude must always be smaller than 10^6 meters.

The divide check light is tested upon entrance into the subroutine. If it is on at this time, it is turned off and a message is printed. Computation then proceeds. The divide check light is again tested prior to exit from the subroutine, and if it is on a message will be printed. Computation is then halted and EXIT is called.

Required Subroutines

The subroutines ATAN, SQRT, EXP, and ALOG, all of which are on the FORTRAN IV library tape, are used by AL ARDC (ARDC1)

Method

The tabulated functions given by the ARDC Model Atmosphere, 1959, are approximated over specific altitude ranges by various polynomials and other functions.

References

Minzner, R. A., Champion, K.S.W., and Pond, H.L.:
The ARDC Model Atmosphere, 1959. Air Force Surveys
in Geophysics No. 115 (AFCRC-TR-59-267), Air Force
Cambridge Research Center, August, 1959.

Identification

AL ARTN, Inverse Tangent Function
FORTRAN IV, MAP-coded
Ames Modification of SHARE Library Routine LA S840

Purpose

This subroutine is used to compute the value of the function $Y = \text{Arctan}(X/Z)$, where X , Y , and Z are single-precision floating-point numbers, and Y is in radians.

Usage

This subroutine is entered by use of the statement

$$Y = \text{ARTN}(X, Z)$$

The result will be in the correct quadrant in the range $-\pi$ to $+\pi$.

Special Case

If this subroutine is entered with $X = Z = 0$, the value returned will be zero.

Timing

The average execution time for this subroutine is 362 μs .

Method

A continued fraction approximation is employed in this computation.

Identification

AL ASIN-ACOS, Inverse Sine and Cosine Functions
FORTRAN IV, MAP-coded
Ames Modification of SHARE Library Routine NA 135.3

Purpose

This subroutine is used to compute the value of the functions $Y = \text{Arcsin}(X)$ and $Y = \text{Arccos}(X)$, where X is a single-precision floating-point argument.

Usage

This subroutine is entered by use of arithmetic statements, e.g.,

$$Y = \text{ASIN}(X), -\pi/2 \leq Y \leq \pi/2$$

and

$$Y = \text{ACOS}(X), 0 \leq Y \leq \pi$$

Error Condition

If the absolute value of X is greater than one, an error trace is initiated, and a message is printed out.

Accuracy

The results from the use of this subroutine are accurate to 7 significant figures.

Timing

The average computing time for $\text{ASIN}(X)$ and $\text{ACOS}(X)$ are $399 \mu\text{s}$ and $404 \mu\text{s}$ respectively.

Method

A. $\text{Arcsin}(X)$

$$\text{Arcsin}(X) = \pi/2 - \sqrt{1 - X} F(X)$$

$$F(X) = C_0 + C_1X + C_2X^2 + \dots + C_7X^7$$

The constants C_0, C_1, \dots, C_7 are

$C_0 = +1.570796327$	$C_4 = +0.0308918810$
$C_1 = -0.2145988016$	$C_5 = -0.0170881256$
$C_2 = +0.0889789874$	$C_6 = +0.0066700901$
$C_3 = -0.0501743046$	$C_7 = -0.0012624911$

B. $\text{Arccos}(X)$

$$\text{Arccos}(X) = \pi/2 - \text{Arcsin}(X)$$

References

Hastings, Cecil, Jr.: Approximations for Digital Computers.
Princeton University Press, 1955.

Identification

AL BARN, AL RNDM, Random Number Generators
 FORTRAN IV, MAP - coded
 Ames Modifications of SHARE Library Routine BA N203

Purpose

These are two identical subroutines designed to generate a member of either of two sets of random numbers. One set is uniformly distributed on the unit interval, and the other is normally distributed with zero mean and unit standard deviation.

Usage

The statements

$$\begin{aligned} X &= \text{RANDOM} (+1) \\ X &= \text{BARN} (+1) \end{aligned}$$

will generate a random normalized floating-point number which is a member of the set that is uniformly distributed on the unit interval.

The statements

$$\begin{aligned} X &= \text{RANDOM} (-1) \\ X &= \text{BARN} (-1) \end{aligned}$$

will generate a random normalized floating-point number which is a member of the set that is normally distributed with zero mean and unit standard deviation.

Discussion

The random numbers generated by the two subroutines are identical for the same run, and so it is possible to produce two sequences of random numbers by intermixing the use of the two.

Method

The uniformly distributed set of random numbers is generated by the method of congruence. That is, the n th member of the set is

$$X_n = K^n X_0 \text{ Mod } M$$

where

$$\begin{aligned} K &= 991 \\ X_0 &= 321,528,735 \\ M &= 2^{31} - 1 \end{aligned}$$

The mean of this set is

$$\mu_X = \int_0^1 X dX = 1/2$$

and the variance is

$$\sigma_X^2 = \int_0^1 (X - 1/2)^2 dX = 1/12$$

If \bar{X} is the mean of any selection of n of the uniform numbers, the Central Limit Theorem states that the variable \bar{X} approaches a normal distribution where n is sufficiently large. Also,

$$\mu_{\bar{X}} = \mu_X = 1/2$$

and

$$\sigma_{\bar{X}}^2 = \frac{\sigma_X^2}{n} = \frac{1}{12n}$$

The random variable

$$Y = \sqrt{12n} (\bar{X} - 1/2)$$

is normally distributed with $\mu_Y = 0$ and $\sigma_Y^2 = 1$. A satisfactory value of n is 20, for which

$$Y = \sqrt{240} \left[\sum_{i=1}^{20} \frac{X_i}{20} - \frac{1}{2} \right]$$

Thus, the normally distributed set is constructed from the uniform set.

References

- Johnson, D. L.: Generating and Testing Random Numbers on the IBM Type 701.
 Boeing Airplane Company Document D1-6717, March 8, 1955.
 Lehmer, D.A.: Mathematical Methods in Large-Scale Computing Units.
 Proceedings of the Second Symposium on Large-Scale Digital
 Calculating Machinery, Harvard University Press, Cambridge, Mass., 1951.

Identification

AL BNIN, Subroutine to Read Binary Information From Tape
 FORTRAN IV, MAP-coded
 Written by B. L. Meyer

Purpose

This subroutine is used to read binary physical records of arbitrary length from magnetic tape.

Usage

The calling statement is

CAL BNIN(LTN,A,N,I)

where

LTN	is the logical tape number
A	is an array into which binary records are read
N	is the number of 36-bit binary words to be read from tape in a single record
I	is an error return such that
	I = 0 for normal return
	= 1 for EOF on tape
	= 2 for redundant record on tape

Discussion

Upon execution of the CALL statement, a binary record is read into array A. This array must be of dimension at least N. There is no restriction on the size of binary records that may be read with the use of this subroutine.

Restriction

The program that uses AL BNIN should not contain any FORTRAN tape manipulation statements for the tape unit referenced by AL BNIN. A special subroutine, AL NDID, has been prepared for the special purpose of rewinding and backspacing these tapes.

AL LOCA may be used with these tapes, also. .

References

AL LOCA	Ames FORTRAN IV Library Subroutine
AL NDID	Ames FORTRAN IV Library Subroutine.

Identification

AL CLOK (CLOCK), Access the Computer-Controlled Clock
FORTRAN IV, MAP-coded

Purpose

This program provides access to the computer-controlled clock at any point in a program.

Usage

The clock is accessed by the statement

CALL CLOCK(X)

where X is the time, in minutes, from an arbitrary reference. This time is accurate to a sixtieth of a second.

Timing

The access time to the clock, using this subroutine, is approximately .227 ms.

References

This program is based on information in CEI31, IBM Special Systems Bulletin "IBM Program Accounting Clock, RPQ78054".

Identification

AL COSH, SINH, Hyperbolic Sine and Cosine Function

See writeup for FORTRAN IV Library Routine AL SINH.

Identification

AL COUN (COUNT), Counting Routine for Use with AL DDMP
FORTRAN IV, MAP-coded
Written by V. L. Sorensen

Purpose

This routine is used in conjunction with the program AL DDMP in order to be able to execute a CALL DDUMP every K-th time, rather than every single time.

Usage

The subroutine COUNT is called by use of the statement

CALL COUNT(I,J,K)

which must immediately precede the CALL DDUMP statement that is to be controlled. The CALL DDUMP will be executed the I-th time, and every K-th time thereafter until J is exceeded.

Identification

AL CRSH (CRASH), Post-Mortem Dumping Routine

The program AL CRSH is not on the Library tape, and must be requested by use of a \$DECK Control card. The library tape contains a dummy procedure, however, so that CALL CRASH statements need not be removed from FORTRAN IV decks.

Writeups for AL CRSH are available in the CAB library files.

Identification

AL COWL, Numerical Integration of Second Order Differential Equations
by the Cowell Method

FORTRAN IV

Written by M. M. Resnikoff

Purpose

This subroutine is used to obtain the numerical solution of the system of N ordinary differential equations

$$Y_i'' = f_i(X, Y_1, Y_2, \dots, Y_N, Y_1', Y_2', \dots, Y_N'), \quad i=1, 2, \dots, N$$

In these equations, primes indicate differentiation with respect to the independent variable X. The initial conditions are:

$$Y_1(X_0) = (Y_1)_0$$

$$Y_2(X_0) = (Y_2)_0$$

$$\vdots$$

$$Y_N(X_0) = (Y_N)_0$$

$$Y_1'(X_0) = (Y_1')_0$$

$$Y_2'(X_0) = (Y_2')_0$$

$$\vdots$$

$$Y_N'(X_0) = (Y_N')_0$$

Both forward and backward starting procedures are provided

Usage

This subroutine is entered by use of the statement

CALL COWELL(X,T,DERIV, H,M,N)

where

X is the independent variable

T is a two-dimensional array that contains values of Y_i , Y_i' , and Y_i'' (see table below). T must be dimensioned (15,N) in the user's calling program

DERIV is the name of a subroutine, supplied by the user, that computes the second derivatives Y_1'' , Y_2'' , ..., Y_N'' , and stores them in T(3,1), T(3,2), ..., T(3,N). DERIV has arguments X,T.

H is the interval in X over which integration is to take place

M is a special code where

M = -1 backward starter

M = 0 forward starter

M \geq 1 indicates that integration is to be performed from X to X+MH.

N is the number of differential equations in the system.

Note that for M=-1 or 0, only a setup function is performed. The subroutine must be called using one of the two starting procedures before integration can be performed. After each call where integration is specified (M \geq 1), subsequent integration can be carried out without reinitiating a starting procedure. If, however, the interval size H is to be changed, a new start must be made.

The array T is loaded as follows:

$$\left. \begin{array}{l} T(1,1) = Y_1 \\ T(1,2) = Y_2 \\ \vdots \\ T(1,N) = Y_N \\ T(2,1) = Y'_1 \\ T(2,2) = Y'_2 \\ \vdots \\ T(2,N) = Y'_N \\ T(3,1) = Y''_1 \\ T(3,2) = Y''_2 \\ \vdots \\ T(3,N) = Y''_N \end{array} \right\}$$

The user places

initial values

here prior to a

starter entry.

The user's subroutine

DERIV places second

derivatives here

If it is desired to examine the differences ∇f_I , $\nabla^2 f_I, \dots, \nabla^7 f_I$ after any integration step, they may be found in cells $T(4,I)$, $T(5,I)$, ..., $T(10,I)$, where $I = 1, 2, \dots, N$.

The user must name his derivative subroutine (called DERIV herein) in an external statement, and the array T must be explicitly dimensioned $T(15,n)$, where $n = N$

Method

A sixth order Cowell second sum predictor-corrector method is used for continued integration, and an iterated sixth order Cowell second sum method is used for the starters (see reference 1). The predictors are given by the equations

$$\begin{aligned} Y_{n+1}^* &= A_n + H \sum_{k=0}^6 \gamma_{-1,k+1} \nabla^k f_n \\ Y_{n+1}^* &= B_n + H^2 \sum_{k=0}^6 \sigma_{-1,k+2} \nabla^k f_n \end{aligned} \quad (1)$$

$$A_{n+1} = A_n + H f_{n+1}$$

$$B_{n+1} = B_n + H A_{n+1}$$

and the correctors by

$$\begin{aligned} Y_{n+1}^i &= Y_{n+1}^* + H \gamma_{-1,7} \nabla^7 f_{n+1}^* \\ Y_{n+1} &= Y_{n+1}^* + H^2 \sigma_{-1,8} \nabla^7 f_{n+1}^* \end{aligned} \quad (2)$$

where $\gamma_{i,j}$, $\sigma_{i,j}$ are constants.

After Y_{n+1} and Y_{n+1}^i are computed, the function f is again computed and the differences ∇^i are corrected.

The backward starter goes through the following, eight times:

$$p = 1(1) 6$$

$$Y'_{-p} = A_0 + H \sum_{m=0}^6 \gamma_{6,p,m} f_{-m}$$

$$Y_{-p} = B_0 - (p+1)HA_0 + H^2 \sum_{m=0}^6 \sigma_{6,p,m} f_{-m}$$

$$f_{-p} = f(X_{-p}, Y_{-p}, Y'_{-p})$$

(3)

$$A_0 = Y_0 - H \sum_{m=0}^6 \gamma_{6,0,m} f_{-m}$$

$$B_0 = Y_0 + HA_0 - H^2 \sum_{m=0}^6 \sigma_{6,0,m} f_{-m}$$

Initial values are given by

$$A_0 = Y'_0 + \frac{1}{2} H f_0$$

(4)

$$B_0 = Y_0 + H Y'_0 + \frac{5}{12} H^2 f_0$$

The ordinates $Y_0, Y_{-1}, \dots, Y_{-6}$ are then converted to backward differences, $\nabla f_0, \nabla^2 f_0, \dots, \nabla^6 f_0$.

The equations for the backward starter are converted to those for the forward starter by means of the transformation

$$\begin{aligned} H &\longrightarrow -H, & Y'_{-m} &\longrightarrow Y'_m \\ Y_{-m} &\longrightarrow Y_m, & f_{-m} &\longrightarrow f_m \end{aligned}$$

(5)

The resulting ordinates Y_0, Y_1, \dots, Y_6 are then converted to backward differences

$\nabla f_6, \nabla^2 f_6, \dots, \nabla^6 f_6$ at $X=X_0 + 6H$. Extending these differences back to X_0 is accomplished by assuming $\nabla^7 f_6 = 0$

The values of all constants may be found in the previously mentioned NASA TN (reference 1).

References

1. Mersman, Wm. A.: Self-starting Multi-step Methods for the Numerical Integration of Ordinary Differential Equations. NASA TN D 2936, 1965.

Identification

AL CROT, Solution of System of Linear Algebraic Equations Using the Method of Crout

FORTTRAN IV

Written by V. L. Sorensen

Purpose

This subroutine is used to solve the matrix equation $AX = B$, where A is a square coefficient matrix and B is a matrix of constant vectors. The matrix B may consist of several constant vectors.

Usage

The subroutine is entered by use of the statement

CALL CROUT(A,B,N,M,IERR)

where

- A is the array that contains the coefficient matrix
- B is the array that contains the constant column vectors upon entry, and the solution vectors upon return
- N is the order of the coefficient matrix
- M is the number of constant vectors
- IERR is an error return such that

IERR = 0, normal return

IERR = 1, the coefficient matrix is singular

The arrays A and B may be either one or two-dimensional. If they are defined as one-dimensional arrays, then A must be dimensioned at least N^2 and B at least $M * N$ in the calling program. If A and B are to be two-dimensional arrays, then they must be dimensioned exactly $N \times N$ and $N \times M$, respectively, in the calling program. The manner in which the elements of the matrices are to be stored in A and B is illustrated in the example below.

Example

$$x + 3.0y + 6.0z = 10.0, 2.0$$

$$2.0x + 4.0y + 10.0z = 20.0, 1.0$$

$$6.0x + 9.0y + 7.0z = 13.0, 3.0$$

If A and B are two-dimensional arrays, then

$A(1,1) = 1.0$	$A(1,2) = 3.0$	$A(1,3) = 6.0$
$A(2,1) = 2.0$	$A(2,2) = 4.0$	$A(2,3) = 10.0$
$A(3,1) = 6.0$	$A(3,2) = 9.0$	$A(3,3) = 7.0$

and

$B(1,1) = 10.0$	$B(1,2) = 2.0$
$B(2,1) = 20.0$	$B(2,2) = 1.0$
$B(3,1) = 13.0$	$B(3,2) = 3.0$

If A and B are one-dimensional arrays, then

$A(1) = 1.0$	$A(4) = 3.0$	$A(7) = 6.0$
$A(2) = 2.0$	$A(5) = 4.0$	$A(8) = 10.0$
$A(3) = 6.0$	$A(6) = 9.0$	$A(9) = 7.0$

and

$B(1) = 10.0$	$B(4) = 2.0$
$B(2) = 20.0$	$B(5) = 1.0$
$B(3) = 13.0$	$B(6) = 3.0$

The quantities N and M are 3 and 2, respectively.

Method

The method of Crout (see reference 1) is used here. Algorithms described in references 2 and 3 are employed in the programming.

References

1. Nielsen, K. L.: Methods in Numerical Analysis. The Macmillan Co., New York, 1956, pp. 181-188.
2. Thacher, Henry C., Jr.: Crout With Pivoting II, Algorithm 43. Communications of the ACM, vol. 4, no. 6, April 1961, pp. 176-177.
3. Forsythe, George E.: Crout With Pivoting, Algorithm 16. Communications of the ACM, vol. 3, no. 9, Sept. 1960, pp. 507.

Identification

AL CVRT, Subroutine to Convert Data in one Format into Another Format
FORTRAN IV

Written by B. R. Briggs and Joanna L. Phifer

Purpose

This subroutine is used to convert data in a given format into another format. It is useful for processing packed data, and for converting floating point and integer numbers into Hollerith format for use by such plot labelling subroutines as AL PALP.

Usage

The calling statement is

```
CALL CVRT (DATA1,N1,FMT1,DATA2,N2,FMT2)
```

where

DATA1	is the name of the array that contains the original data
N1	is the number of words in array DATA1
FMT1	is the name of the array that contains the format of the original data
DATA2	is the name of the array that contains the converted data
N2	is the number of words in the array DATA2
FMT2	is the name of the array that contains the format for converting the original data

Note: Formats FMT1 and FMT2 may be entered as Hollerith strings in the calling sequence, if desired.

Restriction

The formats FMT1 and FMT2 can treat a maximum of 132 BCD characters (22 fully packed words).

Examples

Example 1. To unpack a binary word. (012 to 3I4)

```
DIMENSION IDATA(3), FMT1(1), FMT2(1)
DATA      WORD/0763102521166/,
1         FMT1(1)/5H(012)/,
2         FMT2(1)/5H(3I4)/
CALL CVRT (WORD,1,FMT1,IDATA,3,FMT2)
```

Example 2. To pack a BCD word. (Integer to Hollerith, 6I1 to A6)

```
DIMENSION INT(6), FMT3(1), FMT4(1)
DATA      (INT(I), I=1,6)/7,6,3,6,5,0/,
1         FMT3(1)/5H(6I1)/,
2         FMT4(1)/4H(A6)/
CALL CVRT (INT,6,FMT3,BCD,1,FMT4)
```

Example 3. To prepare a title. (Real to Hollerith)

```
DIMENSION HEAD(3), FMT5(3), FMT6(1)
DATA      RADIUS/6375.553/,
1         (FMT5(I), I=1,3)/17H(7HRADIUS=,F10.3)/,
2         FMT6(1)/5H(3A6)/
CALL CVRT (RADIUS,1,FMT5,HEAD,3,FMT6)
CALL VTITLE (HEAD,3,0)
```

The above statement illustrates one use for this prepared title. VTITLE is an entry point to subroutine AL PAGE. In FORTRAN IV a similar procedure can be used to prepare variable labels to be printed on plots by subroutine AL PALP.

Example 4. To read data as specified by code letters or numbers on the card (also Hollerith to both Integer and Real by equivalencing "read" array)

```

DIMENSION      IDATA(35), X(8), FMT7(1), FMT8(1), FMT9(2), TITLE(12)
EQUIVALENCE    (NX,X(1))
DATA           FMT7(1)/6H(12A6)/,
1             FMT8(1)/6H(35I2)/,
2             (FMT9(I),I=1,2)/11H(11,7E10.8)/
DO 750        NCARD=1,NCARDS
READ (5,705)  I,TITLE
705  FORMAT (11,11A6,A5)
GO TO (710,720,730), I
710  CALL VTITLE (TITLE,12,0)
      . . . .
GO TO 750
720  CALL CVRT (TITLE,12,FMT7, IDATA, 35, FMT8)
      . . . .
GO TO 750
730  CALL CVRT (TITLE,12,FMT7,X,8,FMT9)
      . . . .
750  CONTINUE
760  . . . .

```

References

AL PAGE, Ames FORTRAN IV Library Subroutine
 AL PALP, Ames FORTRAN IV Library Subroutine

Identification

AL DDMP (DDUMP), Dumping
FORTTRAN IV, MAP-coded
 Written by V. L. Sorensen

Purpose

This subroutine is designed to provide labelled dynamic dumping at execution time of single-precision, double-precision, and complex variables, and Hollerith information.

Usage

The entry is by use of the statement

```
n CALL DDUMP(N, VARLIST, list)
```

where

N is the statement number n
 VARLIST is the name of an array into which labels of variables and arrays are read by use of a DATA statement
 list is the list of names of variables and arrays to be dumped

As implied above, the CALL DDUMP statement has associated with it a DATA statement, by means of which the variable and array labels, and variable type, are loaded into the array VARLIST. The label for a single variable is simply the name of the variable. For an array, on the other hand, the label consists of the array name followed by the limits, in parentheses, of the elements to be dumped from the array. Individual labels are separated, in the DATA statement, by commas. A string of labels preceded by the characters \$C\$ will be interpreted as belonging to a list of complex variables. A string of labels preceded by the characters \$D\$ will be interpreted as belonging to a list of double-precision variables. A string of characters preceded by the symbols \$H\$ will be interpreted as belonging to a Hollerith list. The labels for a real single-precision (floating-point or integer) list should be preceded by the characters \$\$ if other types of labels appear ahead of it in the DATA statement. If no other types of variables are labelled ahead of the single-precision variables, or if only single-precision variables are being dumped, the symbol \$\$ may be deleted. The entire string of characters that comprise the labels is enclosed by parentheses, and the H-format is used to load it into the array VARLIST. Embedded blanks are ignored by the subroutine but should be included in the Hollerith count for the DATA statement. Labels with more than six characters will be truncated.

Discussion

Single variables are printed in sequence, each variable being preceded by its label. Single-precision real (integer and floating-point) variables in arrays are printed 7 values per line, and double-precision, complex variables in arrays appear 4 values per line, preceded by the array name and the indices of the elements on that line. The format with which a variable is printed is chosen according to the following rules:

1. If a single-precision variable looks like a non-normalized floating-point number, or if the exponent is zero, the variable is presumed to be an integer and is printed out using the I-format. Otherwise E-format is used.
2. A complex variable is always printed in the E-format, the real part first, followed by a comma, and then the imaginary part.
3. A double-precision variable is printed using the D-format and 16 decimal places.
4. A Hollerith variable is printed using the A-format.

Multidimensional arrays are treated as though they were one-dimensional, and elements of a multidimensional array that are to be dumped must be located by the following rule:

The element (I,J,K) of an array A, which is of dimension (L,M,N), is found as the NN-th element of a one-dimensional array by the use of the formula

$$NN = I + (J - 1)L + (K - 1)LM$$

As an example, suppose that it is desired to dump elements (2,3,4) through (6,7,8) of an array that has been dimensioned (10,10,10). Use of the above formula leads to the result that this range of elements is equivalent to the range (322-766) of the corresponding one-dimensional array.

Restriction

There is a limit of 1000 lines of DDUMP output. If more than this limit is requested, the portion beyond 1000 lines is automatically deleted.

Examples

The first example illustrates the coding for dumping single-precision variables A,B, and K, and elements 5 through 25 of single-precision array D.

```

      DIMENSION VARLST(3)
      DATA(VARLST(I),I = 1,3)/15H(A,B,K,D(5-25))/
n     CALL DDUMP(N,VARLST,A,B,K,D)

```

Note that the symbol \$\$ is not needed, since only single-precision variables are being dumped. The items in the CALL list correspond one-to-one with the labels in the DATA statement.

The next example illustrates all of the features of the dumping subroutine.

```

      DOUBLE PRECISION DOUB,DOUBA
      COMPLEX COMP,COMPA
      DIMENSION VARLST(17),REALA(20),INTA(5),HOLLA(5),
1    COMPA(10),DOUBA(5)
      DATA(VARLST(I),I=1,17)/102H(REAL,RE ALA(2-10),INT,
1    INTA(5)$H$HOL, HOLLA(1-2),$C$COMP,COMPA(1-10),$RE
2    ALL$D$DOUB,DOUBA(1- 5)$ETC)/
100   CALL DDUMP(100,VARLST,REAL,REALA,INT,INTA,
1    HOLL,HOLLA,COMP,COMPA,REAL 1,DOUB,DOUBA,ETC)

```

Notice that single variables and arrays of all types are dumped. Observe also that in the first continuation card of the DATA statement a comma has not been placed ahead of the \$H\$ symbol, but one does, in fact, appear ahead of the symbol \$C\$. This merely illustrates that the comma is optional here. Note the blanks, also.

Notes

The programmer should be aware that unused cells are not set to zero in FORTRAN IV but are loaded with a particular code number that is constant within a given program. Cells containing this code number are not dumped, but the words NO VAL are printed in place of the number.

The subroutine AL COUN, discussed elsewhere has been prepared for use with AL DDMP

Identification

AL DE6F (DE6FN), Cowell Integration of 2nd Order Differential Equations
 FORTRAN IV, MAP-Coded Subroutine
 Ames Modification of SHARE Library Routine RW DE6F

Purpose

The purpose of this subroutine is to obtain a numerical solution of the system of N ($N \geq 1$) ordinary second-order differential equations

$$(1) \quad \begin{cases} y_1'' = f_1(x, y_1, y_2, \dots, y_N, y_1', y_2', \dots, y_N') \\ y_2'' = f_2(x, y_1, y_2, \dots, y_N, y_1', y_2', \dots, y_N') \\ \vdots \\ y_N'' = f_N(x, y_1, y_2, \dots, y_N, y_1', y_2', \dots, y_N'), \end{cases}$$

or the system

$$(2) \quad \begin{cases} y_1'' = f_1(x, y_1, y_2, \dots, y_N) \\ \vdots \\ y_N'' = f_N(x, y_1, y_2, \dots, y_N), \end{cases}$$

with first derivatives missing,

where x is the independent variable, and with the initial conditions

$$(3) \quad \begin{cases} y_1(x_0) = (y_1)_0 \\ \vdots \\ y_N(x_0) = (y_N)_0 \\ y_1'(x_0) = (y_1')_0 \\ \vdots \\ y_N'(x_0) = (y_N')_0 \end{cases}$$

where

$$y'_j \equiv \frac{dy_j}{dx}, \quad y'' \equiv \frac{d^2y_j}{dx^2} \quad (j = 1, 2, \dots, N).$$

The solutions are obtained for discrete values of the independent variable x at specified intervals, that is, from x_i to x_{i+1} ($i = 0, 1, 2, \dots$). Each interval h is given by

$$h = x_{i+1} - x_i.$$

Usage

The subroutine has twelve entries. The first two must be used with all problems, but the other ten may be used optionally.

1. Set-Up Entry

When this entry is used the subroutine does all necessary initialization to start the integration at $x = x_0$. It then enters the derivative subroutine, which is supplied by the user, to obtain $y''_j(x_0)$ ($j = 1, 2, \dots, N$) and returns control to the main program.

This entry must be used whenever it becomes necessary to restart the integration at some intermediate point $x = x_I$ such as a point of discontinuity. In this case x_0 is set to x_I . The set-up entry is also used whenever the entry for a running start (see entry number 3 below) is used.

The CALL statement, with normal conventions for integer and floating-point number designations, is:

```
CALL DE6FN(IDR,IOP,T,N,DERIV,JN,KR,SD,HMIN,HMAX,YMIN),
```

where

IDR	indicates whether first derivatives are present or not (i.e., whether system (1) or (2) is to be solved) as follows: IDR = +1 if first derivatives are present. IDR = -1 if first derivatives are not present.
IOP	indicates the desired mode of operation (See Method). IOP = +1 if variable step-size mode is to be used. IOP = -1 if fixed step-size mode is to be used.
T	is an array containing $30N + 3$ cells and is reserved by the user.

- N** is the number of differential equations in system (1) or (2).
- DERIV** is the name of a FORTRAN subroutine (supplied by the user) that evaluates the second derivatives y'' ($j = 1, 2, \dots, N$) and stores them in $T(4 + 2N)$ through $T(4 + 3N-1)$. The name of the subroutine DERIV must appear on an EXTERNAL card in the calling program. This subroutine has no arguments.
- JN** ($\leq N$) is the number of equations (in System (1) or (2)) that should be used when DE6FN is testing for a change in interval size (See Method). If $JN = 0$ the routine sets it to $JN = N$.
- KR** is the ratio of a Cowell step to a Runge-Kutta step and is used in starting the integration (See Method). If $KR = 0$ the routine sets it to $KR = 4$.
- SD** is equivalent to the number of significant digits required at each integration step (See Method). If $SD = 0$ the routine sets it to $SD = 1.0E-9$.
- HMIN** is the minimum value of $|h|$ below which the subroutine will not decrease $|h|$ (See Method).
- HMAX** is the maximum value of $|h|$ above which the subroutine will not increase $|h|$ (See Method). If $HMAX = 0$ the routine sets it to $HMAX = 1.0E+18$.
- YMIN** is the minimum value of y to use in controlling changes of interval size. If $YMIN = 0$ the routine sets $YMIN = 1.0$ (See Method).

If the fixed mode of operation (i.e., if $IOP = -1$) is used, then SD , $HMIN$, $HMAX$ and $YMIN$ are arbitrary, but the user should set $JN = 1$ for maximum efficiency.

The array T and the integer N should appear in COMMON since they are referred to in both the calling program and the subroutine that evaluates the second derivatives. Any other parameters needed in the derivative subroutine should be transmitted through COMMON.

The array T contains the following information:

$T(2) = x,$	the independent variable
$T(3) = h,$	
$T(4) = y_1(x)$	the value of the interval size
$T(5) = y_2(x)$	
.	values of the dependent variables
.	
.	
$T(4+N-1) = y_N(x)$	values of the first derivatives
$T(4+N) = y_1'(x)$	
$T(4+N+1) = y_2'(x)$	
.	values of the second derivatives which are supplied by the sub-routine DERIV
.	
.	
$T(4+2N-1) = y_N'(x)$	values of the second derivatives which are supplied by the sub-routine DERIV
$T(4+2N) = y_1''(x)$	
$T(4+2N+1) = y_2''(x)$	
.	
.	
.	
$T(4+3N-1) = y_N''(x)$	

Prior to the Set-Up entry the user must set $T(2) = x_0$, $T(3) = h$, $T(4)$ through $T(4+N-1)$ to $y_j(x_0)$, and, if needed, $T(4+N)$ through $T(4+2N-1)$ to $y_j'(x_0)$ ($j = 1, 2, \dots, N$). $T(1)$ is not available to the user.

2. Integration Entry

This entry is used to integrate one step (i.e., from x_i to x_{i+1}). The CALL statement for this entry is:

CALL DE6FP1(ACCUM).

When control is returned to the main program the solutions $(y_1)_{i+1}, (y_2)_{i+1}, \dots, (y_N)_{i+1}$ will be located in $T(4)$ through $T(4+N-1)$, the first derivatives $(y_1')_{i+1}, (y_2')_{i+1}, \dots, (y_N')_{i+1}$, if required, will be located in $T(4+N)$ through $T(4+2N-1)$, and the second derivatives will be in $T(4+2N)$ through $T(4+3N-1)$, where $i = 0, 1, 2, \dots$ ($(y_j)_i = y_j(x_i)$). x_{i+1} will be located in $T(2)$.

The variable ACCUM indicates whether the integration step just performed was a Cowell integration or a Runge-Kutta integration. This information is needed only when a running change of coordinates is made (See Method and entry number 4)

ACCUM < 0 if a Cowell step was just performed

ACCUM > 0 if a Runge-Kutta step was just performed

3. Entry for a Running Start

This entry is used when the user has available starting values for the solutions y_j and, if needed, their derivatives y'_j ($j=1, 2, \dots, N$), or good approximations to them, for eight consecutive values of the independent variable for a fixed step-size h . With this entry, the above values will be used in place of the starting procedure that is incorporated in the subroutine.

To use this entry, the user must first initialize the T array as described under the Set-Up entry. Store $x = x_0 + 7h$ in T(2), where x_0 is the first point in the sequence of eight points. The values of y_j and y'_j must correspond to this value of x . Next, execute the Set-Up entry. Then, store eight consecutive sets of y_j for a fixed value of h starting at cell T(4 + 11N) and eight consecutive sets of y'_j , if needed, for the same fixed value of h (begin with values that correspond to x_0) starting at cell T(4 + 3N) and execute the following CALL statement one time:

CALL DE6FP2

This causes the subroutine to compute the difference table, integrate over one Cowell step (see Method), and advance x from $x_0 + 7h$ to $x_0 + 8h$. Continue with the Integration Entry (entry number 2) to integrate further.

4. Entry for Running Change of Coordinates

Sometimes the user may want to alter or adjust the units on the results beginning at some intermediate point of the integration. At the same time, he may also want to change the derivative routine and/or the units on x and h .

The user must observe the following procedure in order to make the above changes:

After a Cowell integration step (see entry number 2), initialize the beginning of a change of coordinates by executing the CALL statement

CALL DE6FNZ (ANZ),

where ANZ is any non-zero number. Continue to use the Integration Entry (entry number 2) and, starting with the present values, begin to store eight consecutive sets of the solutions $y_j(x)$ beginning at cell $T(4 + 11N)$ and, if needed, eight consecutive sets of the derivatives $y_j'(x)$ beginning at cell $T(4 + 3N)$ ($j = 1, 2, \dots, N$). When the eighth set of values has been stored, any necessary adjustment can be made on the data. The integration may then be continued by using the Integration Entry (entry number 2).

If the subroutine is operating in the variable step-size mode and h is to be altered, then entry number 7 (Change from Variable Mode to Fixed Mode entry) followed by entry number 5 (Change of Step-Size entry) must be used before initializing a change of coordinates (i.e., before CALL DE6FNZ (ANZ) statement). After the change of coordinates has been completed, the variable mode may be restored at any time, if desired, by using entry number 8 (Change from Fixed Mode to Variable Mode entry).

5. Entry for Change of Step-Size

Any time the step-size h is to be changed, use the CALL statement.

CALL DE6FP3(H),

where H is the new value of the step-size. A Runge-Kutta integration step is performed with this entry.

6. Entry for Change of Step-Size for A Final Integration

This entry is used whenever the integration is to be ended at a specific value of x and the value of h is such that this cannot be done with a normal integration entry (entry number 2). Set $T(3)$ to the required value of h and execute the CALL statement

CALL DE6FP4.

This causes one Runge-Kutta integration step to be performed. This entry may also be used at some intermediate value of x if entry number 5 is used immediately afterwards.

7. Entry for Change from Variable Mode to Fixed Mode

CALL DE6FNG

8. Entry for Change from Fixed Mode to Variable Mode

CALL DE6FPS

9. Entry for Change of h_{\min}

CALL DE6FMN (HMIN),

where HMIN is the new value of h_{\min} .

10. Entry for Change of h_{\max}

CALL DE6FMA(HMAX),

where HMAX is the new value of h_{\max} .

11. Entry for Change of y_{\min}

CALL DE6FCM(YMIN),

where YMIN is the new value of y_{\min} .

12. Entry for Change of KR

After any Cowell integration step, use

CALL DE6FCH(KR,R),

where R is the floating-point value of KR (see Set-Up Entry for definition of KR).

Notation

For simplicity of notation, let $Y(X)$, $Y'(X)$ and $F(X, Y, Y')$ be N -dimensioned vectors, the elements of which are y_1, y_2, \dots, y_N ,

y'_1, y'_2, \dots, y'_N and f_1, f_2, \dots, f_N respectively. Thus,

$Y(X) = (y_1, y_2, \dots, y_N)$, $Y'(X) = (y'_1, y'_2, \dots, y'_N)$ and

$F(X, Y, Y') = (f_1, f_2, \dots, f_N)$. The initial conditions can

be designated as Y_0 and Y'_0 ; i.e.,

$Y_0 = (y_1(x_0), y_2(x_0), \dots, y_N(x_0)) = Y(x_0)$ and

$Y'_0 = (y'_1(x_0), y'_2(x_0), \dots, y'_N(x_0)) = Y'(x_0)$. The system (1)

and initial conditions (3) may then be written

$$(4) \quad \begin{cases} Y'' = F(X, Y, Y') \\ Y(x_0) = Y_0 \\ Y'(x_0) = Y'_0 \end{cases}$$

and the system (2) with initial conditions may be written

$$(5) \quad \begin{cases} Y'' = F(x, Y) \\ Y(x_0) = Y_0 \\ Y'(x_0) = Y'_0 \end{cases}$$

The symbol Y_j is taken to mean a solution of (4) or (5) that

corresponds to some value x_j of the independent variable x ;

i.e., $Y_j = Y(x_j)$. Similarly,

$$Y'_j = Y'(x_j)$$

$$Y'' = F_j = F(x_j, Y_j, Y'_j) \text{ or}$$

$F_j = F(x_j, Y_j)$ if first derivatives are missing ($j = 0, 1, 2, \dots$).

The interval size h (or step-size, or step) is given by

$$h = x_{j+1} - x_j$$

Method

The subroutine uses a sixth order Cowell first and second sum predictor-corrector method to integrate at each step. The classical fourth-order Runge-Kutta method is used to obtain starting values. Either of two standard modes of operation may be used in the integration; a fixed-step mode or a variable-step mode. The Cowell formulas are given below:

A. Predictor Formulas

$$(6) \quad pY_j = h^2 S_{j+1}^{II} + h^2 \sum_{k=0}^6 N_k \nabla^k F_{j-1}$$

$$(7) \quad pY'_j = h S_j^I + h \sum_{k=0}^6 M_k \nabla^k F_{j-1} ,$$

where ∇^k is the k th backward difference operator, N and M are constants (Constant List), p is used to indicate "predicted value", S^I and S^{II} are the first and second sums and are defined in equations (8) and (9) ($j = 7, 8, \dots$).

$$(8) \quad S_j^{II} = \frac{Y_{j-1}}{h^2} - \sum_{k=0}^3 C_{2k} \nabla^{2k} F_{j-1}$$

$$(9) \quad S_j^I = \frac{Y'_{j-1}}{h} - \sum_{k=0}^6 D_k \nabla^k F_{j-1} ,$$

where C and D are constants ($j = 7, 8, \dots$).

Note: Backward differences are defined by

$$\nabla F_j = F_j - F_{j-1}$$

$$\nabla^2 F_j = \nabla(\nabla F_j) = \nabla F_j - \nabla F_{j-1}$$

⋮

$$\nabla^m F_j = \nabla(\nabla^{m-1} F_j)$$

B. Corrector Formulas

$$(10) \quad cY_j = h^2 S_{j+1}^{II} + h^2 \sum_{k=0}^6 B_k \nabla^k pF_j$$

$$(11) \quad cY'_j = h S_j^{II} + h \sum_{k=0}^6 G_k \nabla^k pF_j$$

where B and G are constants and c indicates "corrected value"

(j = 7, 8, ...).

C. Integration Procedure

The fourth-order Runge-Kutta method is used to calculate $Y_1, Y_2, \dots, Y_6, Y'_1, Y'_2, \dots, Y'_6$ (that is, the first six entries for integration (entry number 2 under Usage) use Runge-Kutta) with a step-size of h/KR . The integer KR allows Runge-Kutta to operate at a smaller step than the Cowell formulas. If KR is unspecified in the Set-Up Entry it is set to $KR = 4$. After each integration step the derivative subroutine DERIV is entered to obtain F_1, \dots, F_6 .

After Y_6, Y'_6 and F_6 have been calculated a difference table is constructed and stored. When the seventh integration step is called for (i.e., seventh entry), the routine first calculates S_8^{II} and S_7^{II} . It then computes pY_7, pY'_7 and enters DERIV for pF_7 . It then calculates cY_7, cY'_7 and enters DERIV for cF_7 all of which it takes as the final values for the integration (thus; $Y_7 = cY_7, Y'_7 = cY'_7$ and $F_7 = cF_7$). The difference table is adjusted to account for the new integration and control is returned to the calling program. The steps taken for Y_j, Y'_j and F_j ($j = 8, 9, \dots$), are the same as those taken for Y_7, Y'_7 and F_7 except when the variable step-size mode is in operation.

D. Variable Step-Size Mode of Operation

In this mode of operation the difference table is constructed on the sixth entry, as before, but on the seventh entry h is tested

before proceeding to a Cowell step (equations (6) through (11)). Only the first JN ($\leq N$) equations are used in testing h and if JN is unspecified in the Set-Up Entry the routine sets it to JN=N. If JN is other than N, the user should list the equations in order of decreasing importance in order to take full advantage of the variable step-size mode.

The step-size h is halved, left as it is, or doubled in accordance with the following inequalities:

$$\text{if } V \geq \frac{10^3(SD)}{h^2}, \text{ h is halved;}$$

$$\text{if } \frac{10^{-1}(SD)}{h^2} < V < \frac{10^3(SD)}{h^2}, \text{ h is left as it is;}$$

$$\text{if } V \leq \frac{10^{-1}(SD)}{h^2}, \text{ and } W \leq \frac{10^{-1}(SD)}{h^2}, \text{ h is doubled;}$$

where SD is equivalent to the number of decimal places to be retained at each integration step (SD = 1.0E-9 if it is unspecified in the Set-Up Entry),

$$V = \max \left| \frac{\nabla^5 F_j}{\max(Y_j, Y_{\min})} \right|$$

$$W = \max \left| \frac{\nabla^6 F_j}{\max(Y_j, Y_{\min})} \right| \quad (j = 6, 7, 8, \dots),$$

where y_{\min} (= YMIN in Set-Up Entry) is the minimum value to use in calculating V and W and avoids division by unnecessarily small number ($y_{\min} = 1.0$ if unspecified in Set-Up Entry), and where maximums are taken with respect to the first JN equations in system (1) or (2).

If h is left as it is, the routine proceeds to integrate using the Cowell equations. Otherwise, h is halved or doubled and Runge-Kutta integration takes place for six more entries and the above procedure is repeated. If h_{\min} (=HMIN) and h_{\max} (=HMAX) are specified, then h will be altered so that it is always $h_{\min} \leq h \leq h_{\max}$. If they are unspecified, then they are set to $h_{\min} = 0.0$ and $h_{\max} = 10^{18}$.

NOTE: If it is indicated in the Set-Up Entry that first derivatives are not present in the differential equations, then any equation in DE6FN that calculates or uses first derivatives will NOT be computed. If the user wants the first derivatives made available, then it should be specified in the Set-Up Entry that they are present in the differential equations regardless of whether system (1) or (2) is being solved.

E. Running Start

When a running start is used to begin the integration, the user must supply the values $Y_0, Y_1, \dots, Y_7, Y'_0, Y'_1, \dots, Y'_7, h$ and $x_0 + 7h$, where h is fixed. The values Y_j and Y'_j ($j = 0, 1, \dots, 7$) must correspond to $x_0 + jh$.

After these values have been supplied, and entry number three has been executed, the subroutine bypasses the Runge-Kutta starting procedure, constructs the difference table, and performs one Cowell integration step. Thereafter, integration proceeds as described above. A running start may be used at any point x_I , where x_I replaces x_0 , by using the above procedure.

F. Running Change of Coordinates

When the user has accurate information available about the behavior of the solutions of the differential equations, a running change of coordinates may be used to adjust the data to conform to this behavior. If it is desired to change the derivative subroutine or the units on any of the data (i.e., on X, h, Y_j and/or Y'_j), then a running change of coordinates must be made.

The procedure is nearly the same as in a running start (see entry number 4 under Usage) except that the user must save eight consecutive sets of values of Y_j and Y'_j while operating under the fixed step-size mode. After the eight sets have been saved, any necessary adjustment may be made. Integration then continues normally. If the program was operating in the variable step-size mode prior to a change of coordinates, this mode will have to be restored, if desired, before continuing the integration.

G. Other Provisions

In addition to above procedures, the subroutine has provisions for

- a. changing the step-size h at any time (the Runge-Kutta procedure is used for six points and the difference table is adjusted to the new value of h any time h is changed),

- b. changing the step-size for a final integration (to make the integration end at a specific value of x), or to integrate to a specific value of x (see entry number 6 under Usage),
- c. changing from the variable step-size mode to the fixed step-size mode and vice-versa,
- d. and for changing h_{\min} , h_{\max} , y_{\min} and KR.

H. Constants

The following constants are used in the Cowell equations
(equations (6) - (11)):

$N_0 = 0.0833333333$	$M_0 = 0.5$
$N_1 = 0.0833333333$	$M_1 = 0.416666666$
$N_2 = 0.0791666667$	$M_2 = 0.375$
$N_3 = 0.075$	$M_3 = 0.348611111$
$N_4 = 0.07134588948$	$M_4 = 0.329861111$
$N_5 = 0.0620436508$	$M_5 = 0.315591931$
$N_6 = 0.06549575617$	$M_6 = 0.304224537$
$C_0 = 0.0833333333$	$D_0 = -0.5$
$C_2 = -0.00416666667$	$D_1 = -0.0833333333$
$C_4 = 0.0005125661376$	$D_2 = 0.041666667$
$C_6 = 0.00007964065256$	$D_3 = 0.0152777778$
	$D_4 = -0.00768888889$
	$D_5 = -0.00315806883$
	$D_6 = 0.00157903443$
$B_0 = 0.0833333333$	$G_0 = 0.5$
$B_1 = 0.0$	$G_1 = 0.0833333333$
$B_2 = 0.00416666667$	$G_2 = 0.0416666667$
$B_3 = 0.00416666667$	$G_3 = -0.02638888889$
$B_4 = -0.003654100529$	$G_4 = -0.01875$
$B_5 = -0.003141534392$	$G_5 = -0.01426917989$
$B_6 = -0.002708608907$	$G_6 = 0.0833333333$

Identification

AL DIF3, Smooth and Differentiate Unequally Spaced Data
 FORTRAN IV, MAP-coded
 Ames Modification of SHARE Library Routine CL SMD3

Purpose

The purpose of this subroutine is to smooth $N(N \geq 7)$ data points, which may be unequally spaced, by the method of least squares. Options to differentiate and to minimize random errors (i.e., to discard "wild" points) are provided.

Usage

The subroutine AL DIF3 is entered by use of the CALL statement

```
CALL DIF3(X,Y,IT,N,M,IS,J)
```

where

X is the array of abscissas (dimension N)

Y is the array of ordinates (dimension N)

IT is an option word:

IT = 0 if only the smoothing operation is to be carried out

IT = 1 if wild points are to be discarded, followed by smoothing

IT = 2 if smoothing is to be followed by differentiation

IT = 3 if wild points are to be discarded, followed by smoothing, followed by differentiation

N is the number of data points (≥ 7)

M is the number of times smoothing is to be done

IS is an option word such that if IS = 1, the first point is unaltered by smoothing, and if IS = 0, the first point is permitted to be changed by the smoothing operation

Usage (continued)

J is an error indicator

J = 0 for normal return
 J \neq 0 if N \leq 7, or if an overflow or under-
 flow has occurred

The smoothed, or smoothed and differentiated, results will be found in the array Y upon return from AL DIF3.

Method

Wild points are isolated by the following procedure. A quadratic curve is fitted by the method of least squares to six points. Except for the end points, the six points consist of the three immediately preceding and the three immediately following the point under consideration. For the first three points, the quadratic is fitted through the first seven points, exclusive of the one being considered, and for the last three points it is fitted through the last seven points, exclusive of the point under consideration. A point is considered to be wild if it does not lie within three standard deviations of this quadratic. Wild points are replaced by the corresponding points which lie on the curve.

A data point is smoothed by fitting a quadratic through seven points, namely the point under consideration and the three on either side of it. The point on the curve that corresponds to the point being smoothed is taken to be the smoothed value. The first and last three points are smoothed by noncentral formulas, similar to those used in the wild point screening described above. Derivatives of the smoothed data are obtained by use of a simple three-point formula, where noncentral formulas are employed for the first point and the last point.

Identification

AL DPMU, Real or Complex Roots of a Polynomial with Real Coefficients

FORTTRAN IV

Ames Modification of SHARE Library Routine ML HPRS

See writeup for Ames FORTRAN IV Library Routine ML HPRS.

Identification

AL EDFL (ENDFIL), Continue Computation After All Input Data Has Been Read
FORTRAN IV

Purpose

AL EDFL is a subroutine on the Ames FORTRAN library tape, to which control is passed upon detection of the fact that a given program in execution has read and performed calculations upon all of the given input data. If the programmer does nothing, control is transferred to EXIT, and the job is terminated. If the programmer wishes to continue computation after it has been determined that all input data has been read, he supplies a subroutine, which must be named ENDFIL, to replace the program named ENDFIL that is on the library tape.

Usage

The programmer provides a subroutine named ENDFIL, which performs the desired computation. This subroutine may not have any READ statements, and it must terminate by one of the statements.

CALL EXIT

or

CALL ERROR(I,0)

Identification

AL EROR (ERROR), Indication of Location of an Error
FORTRAN IV, MAP-coded
Written by V. L. Sorensen

Purpose

This subroutine is used to print out the program name and the external formula number at which an error has been detected by the programmer.

Usage

This program is entered by use of the statement

CALL ERROR(I,J)

where

I	is the external formula number, supplied by the programmer, at which the error is detected.
J	is an option word:
J = 0	execution is terminated following the print-out of the error message
J = 1	execution continues following the print-out of the error message

Required Subroutines

This subroutine uses the program EXIT, which is on the library tape.

Identification

AL FPT (·FPT·), Standard Treatment of Overflow and Underflow
 FORTRAN IV, MAP-coded
 Standard IBM 7094 Procedure, Modified by V. L. Sorensen

Purpose

This subroutine provides a standard procedure for the treatment of floating-point arithmetic overflows and underflows. An option is provided which allows the programmer to bypass the standard procedure in order to insert his own routines for testing for overflows.

Standard Treatment of Overflow and Underflow

If the result of a floating-point arithmetic operation is smaller than 10^{-38} (underflow), this result is replaced by a normal zero and computation continues. If the result of a floating-point arithmetic operation is greater than 10^{+38} (overflow), then the message

OVERFLOW AT L IN AC [AND MQ]

is printed, and execution is terminated. Here, L is the location of the command that produced the overflow and AC and MQ represent registers on the computer. The part of the message in square brackets is optional.

In addition to the overflow messages, this routine also is entered when an illegal address is used in conjunction with the double-precision hardware on the computer. The most significant part of a double-precision number and the real part of a complex number must be stored in even locations. The compiler and the loader normally handle this correctly. However in the event of difficulty the message

ADDRESS AT L ODD

is printed and execution is terminated. L is the location at which the error occurs.

Special Usage

If the programmer desires to take special action in the case of overflows, the above standard procedure may be disabled by use of the statement

CALL FXFPT

Overflows which occur following the execution of the above CALL may be detected by use of the statement

CALL OVERFL(J)

where

J = 2	if no overflow has occurred
= 1	if overflow has occurred

and then appropriate action can be taken. The statement

CALL REFPT

will cancel the effect of the previous CALL FXFPT, and overflows will again be treated in the standard way by the subroutine FPT.

Identification

AL GAUS (GAUSS), Integration by the Gauss Quadrature Method (10 point)
 FORTRAN IV, MAP-coded
 Ames Modification of SHARE Library Routine GL GAU2

Purpose

This subroutine is used to compute the integral of a function $F(X)$ over an interval (A, B) .

Usage

The technique for using this subroutine involves a calling sequence of two or more statements, as shown below.

```
(1) DUMMY = GAUSS(A,B,N,X)
(2)
.
.
.
(n) DUMMY = F(X)
```

Here

A is the lower limit of integration

B is the upper limit of integration

N is the number of intervals within the limits
 (A,B) over which the 10 point integration procedure
 is to be applied

X is the variable of integration

The word "DUMMY", which appears in the first and last statements only, is used here to symbolize any floating-point variable whatever, which may be subscripted if desired. The statements that appear between the first and last may be any FORTRAN statements, but control must eventually pass to the final statement, which computes the function to be integrated. After the integral has been calculated, its value will be found in "DUMMY".

Restrictions

The quantities A, B, N, and X may be single or subscripted variables but may never be expressions or constants.

Method

The integral of $F(X)$ is computed by application of a ten-point Gaussian quadrature formula over each of N subintervals within the limits (A, B). Thus, the total integral is the sum of the N parts into which the problem has been divided.

According to the Gaussian method, a set of discrete abscissas a_j , and corresponding weights h_j , are determined such that

$$\int_A^B F(X) dX = \sum_{j=1}^n h_j F(a_j)$$

is as exact as possible for fixed n. If it is stipulated that the result is to be exact if $F(X)$ is a polynomial of degree $2n-1$, then it is found that the a_j are roots of the Legendre polynomial of degree n, and that the weights h_j are found by use of a simple relation. The development of the Gaussian quadrature formula will not be undertaken here, since it can be seen in many standard references such as those cited below. The values for a_j and h_j have been tabulated in these references for various values of n, and as has been stated above, the present usage involves a ten-point ($n = 10$) scheme.

References

Whittaker, Edmund, and Robinson, G.: The Calculus of Observation. Blackie and Son Ltd., London, 1946.

Kopal, Zdenek: Numerical Analysis. John Wiley and Sons, Inc., 1961.

Identification

AL INPT Generalized Data Input Subroutine
 FORTRAN IV, MAP
 General Dynamics

Purpose

This subroutine provides for input of single-precision fixed and floating point numbers, octal numbers, and Hollerith information. Usage is particularly convenient inasmuch as no format statements are required, and data may be loaded in any order irrespective of the order in the calling statement.

Usage

The calling statement is

CALL INPUT (5HALPHA, ALPHA, 4HBETA, BETA,)

In the above, the Hollerith literals represent the external names of variables or arrays as they should appear on data cards. The other arguments are the internal names of the variables and arrays as referenced in the source program. It will become apparent that by using the external names in addition to the symbolic location names, it is possible to enter data for a variable on an input card without regard to its relative location in the calling sequence of the program.

Acceptable Input Data Forms

a) Floating Point
 General Form

Any number of decimal digits, with a decimal point permitted at the beginning, at the end or between two digits. A preceding plus or minus sign is optional. A decimal exponent preceded by E+ or + or - if negative may follow. If no decimal point appears, the exponent is mandatory. The magnitude of the number must be between the approximate limits of 10^{-38} and 10^{38} .

Examples

17.
 5.0
 -.0003
 5.0E3 (5.0×10^3)
 5.0E+3 (5.0×10^3)
 5.0E-7 (5.0×10^{-7})

b) Decimal Integers
 General Form

Any number of digits. A preceding plus or minus sign is optional. Numbers exceeding $2^{35}-1$ will be treated modulo 2^{35} .

Examples

3
 +1
 -28987

c) Octal Integers
 General Form

1 to 12 octal digits. A preceding plus or minus sign is optional. The magnitude of an octal integer may not exceed 377777777777. Representations less than 12 digits will be right-adjusted. If the first digit of a 12-digit representation is 7, it will be interpreted as -3. If this 7 is preceded by a sign,

Examples

\$5
 \$-346
 \$7777777
 \$+0265
 \$-301265543720

- c) cont'd.
an error exit will occur. Thus
77777777777 is valid, but -77777777777
is not. Octal numbers must be preceded
by a \$.

- d) Hollerith Information
General Form

Examples

Any number of characters, including
blanks. The number of characters is
specified by writing nH preceding the
Hollerith information. n is the number
of characters in the block following nH.

14HTHIS IS A TEST
6HALPHA

Rules for Preparation of Data Cards

Blanks are ignored except within Hollerith data fields.

Data must be contained within card columns 1 through 72.

It is not necessary that variable names on the data cards appear in the
same order as those in the calling sequence. The routine will search the
list for the name and its core location.

Individual data items are separated by commas.

An equal sign or a comma separates the name of a variable and its first
data item.

A comma separates the end of a data set and the next variable name.

A data input record is terminated by an asterisk (*).

It is not necessary to input a data set for each name in the calling sequence.

Elements of an array may be skipped by writing consecutive commas--i.e., no
data between the commas; or by singly subscripting the array name. Double
subscripting is illegal. Thus, if it is desired to input data into a
three-element vector V, one could write:

V = 2.79,,1.32

No data would be entered into V(2). What was originally there remains there.
Alternatively, the above could be written:

V(1) = 2.79, V(3) = 1.32

Special Feature

The card image is normally written on the system output unit, tape 6,
prior to being processed by the routine. If an N is punched in
column 73, the card will not be listed. If column 73 contains a C,
the card is treated as a comment only; i.e., it is not scanned for data.
If the card contains CE in columns 73-74, the card will be treated as
a comment card, and a page will be ejected.

Example

If the following call statement appeared in a FORTRAN program,

CALL INPUT (LHA, A, LHB, B, LHC, C, LHD, D, LHP, P, LHR, R, LHS, S)
the input cards could be punched as follows:

A = 3.14159265, B = \$707, C = 1870,	1st card
D = 1., 2., 3., 4., 5., 6., 7., 8., 9.,	2nd card
R(2) = 3, R(5) = \$74., 42,	3rd card
P = 22HTHIS IS A CHECKOUT RUN*	4th card

Note that D must be dimensioned at least 9,
R dimensioned at least 7 and P at least 4.

Also R(1), R(3), R(4), and R(6) are unchanged.

Even though S appears in the CALL statement, it is not necessary that it appear on one of the input cards. The * on card 4 signifies the end of the data record. This means that the routine will return control to the calling program.

Restrictions

The following errors will be detected by the subroutine.

A diagnostic message and the card in error will be printed on the system output unit, tape 6.

1. Name on data card exceeds six characters.
2. Name on data card does not appear in the calling sequence.
3. Punctuation errors.
4. Octal field errors.
5. Decimal or octal data out of range.

This subroutine may not be used for reading double-precision or complex variables.

Identification

AL INT, Adams-Moulton, Runge-Kutta Integration
 FORTRAN IV, MAP-coded
 Ames Modification of SHARE Library Routine RW DE2F

Purpose

The purpose of this subroutine is to obtain a numerical solution of the system of $N(N \geq 1)$ ordinary first-order differential equations

$$(1) \quad \begin{cases} y_1' = f_1(x, y_1, y_2, \dots, y_N) \\ y_2' = f_2(x, y_1, y_2, \dots, y_N) \\ \vdots \\ y_N' = f_N(x, y_1, y_2, \dots, y_N) \end{cases}$$

with x as the independent variable and with the initial conditions

$$(2) \quad \begin{cases} y_1(x_0) = (y_1)_0 \\ \vdots \\ y_N(x_0) = (y_N)_0 \end{cases}$$

and where $y_j' \equiv \frac{dy_j}{dx} \quad (j = 1, 2, \dots, N).$

(Note: The symbol " \equiv " means "is equivalent to".)

Usage

The subroutine has three entries, one for set-up, one to integrate over an interval h , and one to stop the set-up or integration process if an error occurs.

A. Set-Up Entry

When this entry is used, the subroutine does all the necessary initialization to start integrating at $x = x_0$. It then enters the derivative routine, which is supplied by the user, to obtain $y_j(x_0)$ ($j = 1, 2, \dots, N$) and returns control to the main program.

Usage (continued)

This entry must be used to restart the integration at any intermediate point $x = x_I$ such as a point of discontinuity, a point at which the user wishes to change the interval size h or the parameter K (which is defined below). In these cases x_0 is replaced by x_I .

The CALL statement, with normal conventions for integer and floating-point number designations, is:

```
CALL INT(T,N,K,EU,P,A,HMAX,HMIN,BETA,DERIV),
```

where

T is an array of $12N + 3$ cells if Adams-Moulton option is used or $4N + 3$ cells if Runge-Kutta option is used

N is the number of differential equations

$$\begin{cases} = 0 & \text{for Adams-Moulton variable step-size mode} \\ = 1 & \text{for Runge-Kutta mode} \\ = 2 & \text{for Adams-Moulton fixed step-size mode} \end{cases} \quad (\text{See Method})$$

EU is the upper bound of E_{n+1} (See Method) for truncation error testing done in the variable step-size mode ($EU > 0$).

$P(=p)$ is used to compute the lower bound of E_{n+1} and, if used, should be $P > 0$; if $P = 0$ the routine sets $P = 100.0$

A is a constant used to control interval size reduction (See Method) and should be $A > 0$; if $A = 0$ the routine sets $A = 1.0$

$HMAX$ is the maximum value of h beyond which the routine should not increase $|h|$ (if $HMAX = 0$ the routine assumes there is no upper limit on $|h|$)

$HMIN$ is the minimum value of h below which the routine should not decrease $|h|$ (if $HMIN = 0$ the routine assumes there is no lower limit on $|h|$)

$BETA$ is β (See Method), the number used to increase or decrease the interval size and must be $0 \leq \beta \leq 1$ (if $\beta = 0$ the routine sets $\beta = 0.5$)

$DERIV$ is the name of a subroutine (supplied by the user) that evaluates the derivatives in equations (1) and stores them in $T(4 + N)$ through $T(4 + 2N - 1)$; the name of this subroutine must also appear on an EXTERNAL card. This subroutine has no arguments.

If $K = 1$ or $K = 2$ then the quantities EU , P , A , $HMAX$, $HMIN$, and $BETA$ may be specified arbitrarily.

Usage (continued)

The array T contains the following information:

$T(2) = x$, the independent variable	
$T(3) = h$, the value of the interval size	
$T(4) = y_1(x)$	} values of the dependent variables
$T(5) = y_2(x)$	
.	
.	
$T(4 + N - 1) = y_N(x)$	
$T(4 + N) = y'_1(x)$	} values of the derivatives that are supplied by the derivative sub- routine (i.e., DERIV)
$T(4 + N + 1) = y'_2(x)$	
.	
.	
$T(4 + 2N - 1) = y'_N(x)$	

Prior to the Set-Up entry the user must set $T(2) = x_0$, $T(3) = h$ and $T(4)$ through $T(4 + N - 1)$ to $y_j(x_0)$ ($j = 1, 2, \dots, N$).

The parameter N and the array T should appear in COMMON since they are necessarily referred to in both the main program and the subroutine that evaluates the derivatives. The integration subroutine stores N, scaled at 35, in T(1).

B. Integration Entry

This entry is used to integrate one step (i.e., from x_j to x_{j+1}).

When control is returned to the main program the solutions

$(y_1)_{j+1}, (y_2)_{j+1}, \dots, (y_N)_{j+1}$ ($j = 0, 1, 2, \dots$) will be

located in $T(4)$ through $T(4 + N - 1)$ and x_{j+1} will be in $T(2)$.

The derivatives $(y'_1)_{j+1}, (y'_2)_{j+1}, \dots, (y'_N)_{j+1}$ will appear in $T(4 + N)$ through $T(4 + 2N - 1)$.

The CALL statement for this entry is:

CALL INTM

No arguments are required for this statement.

C. Error Entry

This entry is used in the derivative subroutine (DERIV) to terminate set-up or integration when an error occurs. The CALL statement is:

CALL ERINT

Usage (continued)

To take advantage of this entry, a cell in COMMON should be assigned an integer value prior to either a normal or an error return from the derivative subroutine. This cell can then be used in a computed GO TO statement when control is returned to the main program. If this error return is executed, a new set-up entry must be made before any further integration may take place.

When an error entry has been made, the integration or set-up is terminated, and control is returned to the main program at the statement following the CALL statement (either CALL INT() or CALL INIM) that initiated transfer to the derivative subroutine.

CALL ERINT is optional and may be left unused if all the differential equations in (1) are well behaved (i.e., contain no singularities, etc.).

Notation

For simplicity of notation, let $Y(x)$ and $F(x, Y)$ be N -dimensioned vectors, the elements of which are y_1, y_2, \dots, y_N and f_1, f_2, \dots, f_N , respectively. Thus $Y(x) = (y_1, y_2, \dots, y_N)$ and $F(x, Y) = (f_1, f_2, \dots, f_N)$. The initial conditions can be designated as Y_0 ; i.e., $Y_0 = (y_1(x_0), y_2(x_0), \dots, y_N(x_0)) = Y(x_0)$. The system (1) and initial conditions (2) may then be written

$$(3) \quad \begin{cases} Y' = F(x, Y) \\ Y(x_0) = Y_0 \end{cases}$$

The symbol Y_j is taken to mean a solution of (3) that corresponds to some value x_j of the independent variable x . Also,

$$Y_j' = F_j = F(x_j, Y_j). \quad (j = 0, 1, 2, \dots)$$

The interval-size h (or step-size, or step) is given by

$$h = x_{j+1} - x_j.$$

Method

The user has the option of using either of two methods: a fourth-order Runge-Kutta method or a fourth-order Adams-Moulton method. The Adams-Moulton method may be used with either a fixed or a variable step-size mode.

A. Runge-Kutta Method

The classical fourth-order Runge-Kutta method uses the formulas

$$K_1 = hF(x_j, Y_j)$$

$$K_2 = hF(x_j + \frac{1}{2}h, Y_j + \frac{1}{2}K_1)$$

$$K_3 = hF(x_j + \frac{1}{2}h, Y_j + \frac{1}{2}K_2)$$

$$K_4 = hF(x_j + h, Y_j + K_3)$$

$$Y_{j+1} = Y_j + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)$$

$$j = 0, 1, 2, \dots$$

(K_1, K_2, K_3, K_4 are N-dimension vectors).

The subroutine takes advantage of certain computer operations to control the growth of round-off errors in the above formulas.

B. The Adams-Moulton Method

Predictor Formula:

$$Y_{j+1}^{(p)} = Y_j + \frac{h}{24}(55F_j - 59F_{j-1} + 37F_{j-2} - 9F_{j-3}),$$

where the superscript (p) indicates predicted value.

Corrector Formula:

$$Y_{j+1}^{(c)} = Y_j + \frac{h}{24}(9F_{j+1}^{(p)} + 19F_j - 5F_{j-1} + F_{j-2}),$$

where the superscript (c) indicates corrected value; $F_{j+1}^{(p)} = F(x_{j+1}, Y_{j+1}^{(p)})$.

Method (continued)

If the Adams-Moulton option is used, Y_1 , Y_2 , and Y_3 are calculated with the Runge-Kutta formulas (i.e., the first three entries use Runge-Kutta). Thereafter, x_{j+1} ,

$$Y_{j+1}^{(p)}, F_{j+1}^{(p)}, Y_{j+1}^{(c)}, \text{ and } F_{j+1}^{(c)} = F(X_{j+1}, Y_{j+1}^{(c)}) \quad (j = 3, \dots)$$

are computed before control is returned to the calling program, if the fixed-step-size mode is used.

Variable Mode Option

If the variable-step-size mode is used, then the interval is either decreased, left as it is, or increased in accordance with the following inequalities:

If $E_{n+1} \geq EU$ the interval is decreased to βh ;

if $EL \leq E_{n+1} < EU$ the interval size is not altered;

if $E_{n+1} < EL$ for three successive steps, the step-size is increased to $\frac{h}{\beta}$,

where

$$E_{n+1} = \text{Max} \frac{|Y_{n+1}^{(p)} - Y_{n+1}^{(c)}|}{14D} \quad (\text{the maximum is taken with respect to all } y_j \text{ in } Y),$$

$$D = \text{Max} \left(|Y_{n+1}^{(c)}|, A \right) \quad \text{with respect to all } y_j \text{ in } Y$$

EU is an upper bound for E_{n+1} , specified by the user, and is equivalent to specifying the number of significant figures to retain at each integration step (should normally be $10^{-8} \leq EU \leq 10^{-3}$).

$EL = \frac{EU}{p}$ is a lower bound for E_{n+1} (the user specifies p such that $p > 0$)

A is used to prevent unnecessary reduction in h when the magnitudes of the solutions are small (specified by the user such that $A > 0$, the routine will take $A = 1$ unless otherwise specified)

$\beta = 1/2$ unless otherwise specified.

Method (continued)

After an interval is increased, the routine prevents increasing again until six more points have been completed. Decreasing is done as often as necessary.

The user must specify a starting value for h , and may, if desired, specify minimum and maximum values of $|h|$ beyond which the routine will not decrease or increase $|h|$. Negative values of h may be supplied for backward integration.

Identification

AL INVT, Inversion of a square matrix and solution of linear equations
 FORTRAN IV, MAP-coded
 Submitted by Ralph Carmichael

Purpose

This program solves the matrix equation $AX=B$ where A is a square matrix of dimension n and X and B are $n \times m$ matrices ($m \leq n$). An option is provided to calculate the inverse of A. The determinant of A is evaluated.

Usage

The subroutine is entered from a FORTRAN IV program by use of the statement

CALL INVERT (A,N,B,M,NROWS,DET,E)

where

A and B are the given matrices
 N order of the matrix A ($N \leq NROWS$)
 M number of columns in the matrix B
 NROWS number of rows listed in the dimension statement for A
 DET the location where the value of the determinant will be stored
 E an erasable array of dimension $\geq N$ which is used to keep track of the row-column interchanges

Upon return, the inverse matrix A^{-1} will be stored in A and the matrix X will be stored in B. The original A and B matrices are destroyed.

If $M=0$, the matrix A will be inverted but the computation of the X-matrix is omitted. However, there must be an entry for B in the calling sequence. Thus, array B will be left unchanged when $M=0$.

Method

The Gauss-Jordan reduction method is employed.

Timing

The time involved may be estimated from the equation

$$t = C \cdot N^2 (N+M)$$

C is approximately $34 \mu\text{sec}$.

Reference

Ralston and Wilf, Mathematical Methods for Digital Computers, John Wiley, New York 1960, p.43-49.

Identification

AL TTRA, Routine for Finding a Root, Within Specified Bounds, of a Function

See AL TTR2, AL TTRA

Identification

AL ITR2, AL ITRA, Routines for Finding a Root, Within Specified Bounds,
of a Function

FORTRAN IV, MAP-coded

Ames Modification of SHARE Library Routine GM ITR2

Purpose

These subroutines are used to calculate a root of the function $f(X) = 0$.

Usage

The program ITR2 is called by means of the statement

```
CALL ITR (FUNCT,E1,E2,DELX,A,B,X,ERROR)
```

where

FUNCT is the name of a function subprogram that evaluates $f(X)$.

It must be named on an EXTERNAL card

E1 } are numbers used in testing

E2 } for convergence, e.g.,:

If $|X_i| > E1$,

then upon satisfaction of

$$\left| \frac{X_i - X_{i-1}}{X_i} \right| \leq E1$$

the process is terminated.

If $|X_i| \leq E1$,

then upon satisfaction of

$$|X_i - X_{i-1}| \leq E2$$

the process is terminated

DELX is the size of the scanning interval

A } the root is presumed to lie in

B } the interval (A,B)

X the root

ERROR is an error indicator:

ERROR = -1.0 no root can be found

= 0 normal return

= +1.0 if more than 50 iterations have been taken

The subroutine ITRA has been prepared for purposes of nesting.

It is identical in usage and all other respects to ITR2. Its calling statement is

```
CALL ITRA ( )
```

and the quantities within the parentheses are as described for ITR2.

Comments

Any parameters other than X which are required by the subprogram FUNCT may be transmitted from the main program by use of COMMON.

Some caution should be exercised in the selection of the size of DELX, the scanning interval. An excessive amount of computing time is involved when DELX is too small. At the same time, DELX should be sufficiently small that no more than one root be present within this interval.

It may be observed that this subroutine will in most cases find a root, if it exists, in the given interval (A,B). It is, however, less

efficient than the program AL ROOT (C5 1.0, on the FORTRAN IV library tape), although the latter may, on occasion, fail to give desired results.

If the error return code (ERROR) is -1.0, e.g., if 50 iterations have been taken in the search for the root, then the value of X returned may be satisfactory, since its accuracy is of the order of $\text{DELX}/2^{50}$.

Method

The interval-halving method is used to find the desired root. The function $f(X)$ is evaluated at the starting point $X = A$, and at intervals ΔX (DELX) up to and including the endpoint $X = B$. A change of sign of the function across a ΔX interval indicates a possible root in that interval. The interval is halved successively toward $f(X) = 0$ until the desired accuracy is achieved. The function is evaluated only once for each halving step. If 50 halving steps are taken, control is returned to the calling program and an error indication is given. (ERROR = -1.0) The value of X returned may be satisfactory, however, as its error is of the order of $\Delta X/2^{50}$.

Reference

AL ROOT FORTRAN IV Library Routine

Identification

AL LOCA (LOCATE), Position a Magnetic Tape

FORTTRAN IV, MAP-coded

Written by N. A. James and Revised by B. L. Meyer

Purpose

This subroutine permits the programmer to position a magnetic tape at any file relative to the load point or another file.

Usage

The statement

CALL LOCATE(M,N)

will cause tape N to be rewound, following which it will be positioned at file M. The statement

CALL SKIP(M,N)

will cause M files to be skipped (positive M for forward skipping, negative M for backward skipping). If M is -0, the tape will be positioned at the beginning of the present file. The statement

CALL B4MARK(N)

will position tape N at the beginning of the previous end-of-file (EOF) mark.

Error Conditions

1. An error message is printed if the tape number, N, is entered as zero.
2. If M = 0 in the CALL LOCATE(M,N) statement, an error message is printed.
3. If M = +0 in the CALL SKIP(M,N) statement, an error message is printed.

Execution is terminated following the output of any of the above three error messages.

Restriction

Always use LOCATE at least once with SKIP.

Identification

AL LSQP (LSQPOL), Least Squares Polynomial Fit
 FORTRAN IV
 Ames Modification of SHARE Library Routine AN E206

Purpose

This subroutine is used to calculate the coefficients of the polynomials that represent the best least-squares fit to one or more tabulated functions Y of an independent variable X. The variable X has a set of associated weights W. In addition to the polynomial coefficients, the weighted sum of squares, and the error matrix are computed for each function Y.

Usage

The calling statement for LSQPOL is

```
CALL LSQPOL(X,Y,W,RESID,N,SUM,L,A,B,M)
```

where

- X is the array of independent variable values.
- Y is the two-dimensional array of dependent variable values.
- W is the array of corresponding weights. This array should consist of all ones if other weighting is not used.
- RESID is the two-dimensional array where residuals will be stored.
- N is the number of values of X.
- SUM is the array used to store the weighted sums of squares of residuals.
- L is the number of sets of Y values, and is therefore the dimension of the array SUM, and the second dimension of the arrays Y and RESID.
- A is the two-dimensional array used to store the error matrix.

5/10/64

Usage (continued)

- B is the two-dimensional array used to store the polynomial coefficients in order of increasing degree. The second dimension of B is L.
- M is the number of coefficients in the fitted polynomial(s), which are of degree M-1. M therefore determines the second dimension of A.

Restrictions

This subroutine has been compiled on the basis of N=50, and $M \leq 8$, and for compatibility with AL MATINV the first dimension of A and B must be 20. Thus, the user must set up arrays dimensioned as follows: X(50), Y(50,L), W(50), RESID(50,L), SUM(L), A(20,M), B(20,L). If the dimensions of the arrays as compiled in LSQPOL are not adequate in a particular case, the subroutine can be recompiled with revised DIMENSION and COMMON statements. If LSQPOL is to be altered in this fashion, it may be necessary to make changes in the dimensions of arrays in the subroutine MTNV, used internally by LSQPOL.

COMMON has been reserved in both LSQP and MTNV by use of BLOCK COMMON statements. This COMMON area has been labelled "ANE206", and so the latter symbol may not be employed by the user. It is not necessary for the user to reserve any COMMON storage in his program for LSQP or MTNV.

Required Subroutines

The programs AL MTNV, and AL POLY are used internally by AL LSQP. These two programs are on the FORTRAN IV library tape and need not be requested by the user.

Method

The method of least squares involves the minimization of the function

$$F = \sum_{i=1}^N W_i \left[\sum_{j=0}^{M-1} A_j X_i^j - Y_i \right]^2$$

which leads to the linear system of algebraic equations

$$\frac{\partial F}{\partial A_j} = 0, \quad j = 0, 1, 2, \dots, M-1.$$

Method (continued)

The latter system of equations may be expressed as a matrix equation, e.g.,

$$AZ = B.$$

The above system of equations is solved internally by AL LSQP, by use of the subroutine AL MINV, which performs matrix inversion and solves systems of linear equations. The residuals are calculated by subtracting Y_i from the polynomial evaluated at X_i , and the error matrix is simply A^{-1} , the inverse of A.

Identification

AL LSQS Simultaneous solution by least squares of M equations in N unknowns ($M \geq N$)
 FORTRAN IV, MAP-coded
 Ames Modification of SHARE Library Routine CL LSQS3

Purpose

This subroutine is used to solve simultaneously a set of M equations in N unknowns ($M \geq N$). The method of least squares is employed. Residuals and an unbiased estimate of the standard deviation of the solution may also be computed.

Usage

The user has a set of equations of the form

$$X_1 A_{j1} + X_2 A_{j2} + \dots + X_N A_{jN} = P_j, \quad j = 1, 2, \dots, M$$

where $M \geq N$. The solution of this set is obtained by use of the calling statement

CALL LSQSE(A,P,B,X,M,N,IT,SD,RESID,JERR)

where

- A is an array of dimension $M * N$ in which the coefficients A_{ij} are stored:
 $A(1) = A_{11}, A(2) = A_{12}, \dots, A(N) = A_{1N}$
 $A(N + 1) = A_{21}, A(N + 2) = A_{22}, \dots, A(2N) = A_{2N}$
 \vdots
 $A[(M - 1)N + 1] = A_{M1}, A[(M - 1)N + 2] = A_{M2}, \dots, A(MN) = A_{NM}$
- P is an array of dimension M in which the coefficients P_j are stored:
 $P(1) = P_1, P(2) = P_2, \dots, P(M) = P_M$
- B is an array of dimension $N(N + 3)/2$ that is used internally by the subroutine.
- X is an array of dimension N that contains the solution:
 $X(1) = X_1, X(2) = X_2, \dots, X(N) = X_N$
- M is the number of equations
- N is the number of unknowns $X_i, N \leq M$

IT is an indicator that controls the computation of residuals, e.g.,
 IT = 0, residuals are not computed
 IT = 1, residuals are computed

SD is the unbiased estimate of the standard deviation of the fit.

RESID is an array of dimension M + N + 1. The first M cells contain the residuals, and the last N + 1 cells are used internally by the subroutine.

JERR is an error code such that for
 J = +1 an overflow or underflow has occurred
 J = -1 a pivotal element in a determinant is zero
 J = 0 a normal exit has been made.

The dimensioning of arrays has been specified in the preceding discussion. If a system of more than 10 variables ($N > 10$) is to be solved, however, a storage area set aside in the subroutine must be increased in size, thus necessitating a reassembly of the program.

Method

The given set of equations may be written in matrix form, e.g.,

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{M1} & A_{M2} & \dots & A_{MN} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_M \end{bmatrix} \quad (1)$$

or

$$[A] \{X\} = \{P\} \quad (2)$$

where $M \geq N$.

The values of the elements of A and P are usually the result of some type of measurements, and so the above system of equations is, in general, incompatible, that is, numbers X_1, X_2, \dots, X_N cannot be found which will exactly satisfy all M equations. The problem thus becomes one of finding a set of values X_1, X_2, \dots, X_N such that each equation is approximately satisfied and such that the total error is as small as possible.

The method of least squares is used to calculate such a set of values.

The first step in obtaining the desired least squares solution is to pre-multiply both sides of equation (2) by the transpose of $[A]$, namely $[A^T]$ (see ref. 1). That is,

$$[A^T] [A] \{X\} = [A^T] \{P\}$$

or

$$[C] \{X\} = [D]$$

(3)

Upon expansion, equation (3) takes the form

$$\begin{bmatrix} \sum A_{11}^2 & \sum A_{11}A_{12} & \dots & \sum A_{11}A_{1N} \\ \sum A_{12}A_{11} & \sum A_{12}^2 & \dots & \sum A_{12}A_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ \sum A_{1N}A_{11} & \sum A_{1N}A_{12} & \dots & \sum A_{1N}^2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} \sum A_{11}P_1 \\ \sum A_{12}P_1 \\ \vdots \\ \sum A_{1N}P_1 \end{bmatrix} \quad (4)$$

Where the sums are from 1 to M . The matrix $[C]$ is seen to be square, of dimension $N \times N$, and it is shown in the reference that the solution to equation (4) is, in fact, the desired best (least squares) solution. The proof of this is tedious, and will not be reproduced here. The solution to the system represented by equation (4) is obtained as follows. The matrix $[C]$ is triangularized by the method of pivotal condensation. The resulting set of equations is then solved by elimination.

The unbiased estimate of μ_0 , the standard deviation, is given by the relation

$$\mu_0 = \frac{1}{M} \sqrt{\frac{\pi}{2}} \sum_{i=1}^M |R_i| \quad (5)$$

the quantities R_i are the residuals, as computed from the M equations

$$X_1 A_{j1} + X_2 A_{j2} + \dots + X_N A_{jN} - P_j = R_j, \quad j = 1, 2, \dots, M$$

The residuals are stored in the first M cells of the array RESID, and the quantity μ_0 will be found in SD.

It can be shown by a specific example that the solution to equation (4) is indeed the least-squares solution. Consider a three-by-two system:

$$A_{11}X_1 + A_{12}X_2 = P_1$$

$$A_{21}X_1 + A_{22}X_2 = P_2$$

$$A_{31}X_1 + A_{32}X_2 = P_3$$

the set X_1, X_2 that is sought is the one that makes the sum of squares of residuals a minimum, that is to say, the function

$$\begin{aligned} \Phi = & (A_{11}X_1 + A_{12}X_2 - P_1)^2 + (A_{21}X_1 + A_{22}X_2 - P_2)^2 \\ & + (A_{31}X_1 + A_{32}X_2 - P_3)^2 \end{aligned}$$

must be minimized by this set. Then

$$\frac{\partial \Phi}{\partial X_1} = 0, \quad \frac{\partial \Phi}{\partial X_2} = 0$$

lead to the values of X_1, X_2 which minimize the function Φ . Formally,

$$\begin{aligned} \frac{\partial \Phi}{\partial X_1} = & 2[(A_{11}X_1 + A_{12}X_2 - P_1)A_{11} + (A_{21}X_1 + A_{22}X_2 - P_2)A_{21} \\ & + (A_{31}X_1 + A_{32}X_2 - P_3)A_{31}] = 0 \end{aligned}$$

$$\begin{aligned} \frac{\partial \Phi}{\partial X_2} = & 2[(A_{11}X_1 + A_{12}X_2 - P_1)A_{12} + (A_{21}X_1 + A_{22}X_2 - P_2)A_{22} \\ & + (A_{31}X_1 + A_{32}X_2 - P_3)A_{32}] = 0 \end{aligned}$$

These may be condensed into the following form.

$$\begin{aligned} (A_{11}^2 + A_{21}^2 + A_{31}^2)X_1 + (A_{12}A_{11} + A_{22}A_{21} + A_{32}A_{31})X_2 &= A_{11}P_1 + A_{21}P_2 + A_{31}P_3 \\ (A_{11}A_{12} + A_{21}A_{22} + A_{31}A_{32})X_1 + (A_{12}^2 + A_{22}^2 + A_{32}^2)X_2 &= A_{12}P_1 + A_{22}P_2 + A_{32}P_3 \end{aligned}$$

This pair of equations is clearly equivalent to equation (4) in the case of a three-by-two system. The above example can be used as the starting point for a proof by induction that equation (4) is the system whose solution leads to the minimum value for the sum of squares of residuals.

References

Shilov, Georgi E.: An Introduction to the Theory of Linear Spaces (TR by Richard A. Silverman from the original Russian) Prentice-Hall, Inc., 1961.

Identification

AL MINV, Matrix Inversion and Solution of System of Linear Equations

FORTRAN IV

Modification, by Virginia L. Sorensen, of FORTRAN IV

Library Routine AL MTNV

Purpose

This program is used to solve the matrix equation $AX = B$, where A is a square coefficient matrix and B is a matrix of constant vectors. The inverse matrix A^{-1} is obtained, and an option is provided to calculate the inverse matrix and omit solution of the matrix equation. The determinant of A is also evaluated. All elements of matrices, and results, are in single precision.

Usage

This subroutine is entered by use of the statement

CALL MINV(A,N,B,M,DET)

where

A is an array, dimensioned exactly $N \times N$ in the calling program, wherein the elements of the matrix A are stored by columns.

N is the order of matrix A

B is an array, dimensioned exactly $N \times M$ in the calling program, wherein the constant column vectors B are stored.

M is the number of column vectors in matrix B

DET is the location where the value of the determinant of A is to be placed.

Upon return, the inverse matrix A^{-1} will be found in the array A , and the solution vector X will be found in array B . If the subroutine is entered with $M = 0$, then only the inverse matrix A^{-1} will be computed.

Precaution

The arrays A and B are two-dimensional, dimensioned exactly $N \times N$ and $N \times M$, respectively. Alternatively, they may be one-dimensional arrays, solidly loaded.

Method

Jordan's method is used to reduce the matrix A to the identity matrix I through a succession of elementary transformations, $T_n T_{n-1} \dots T_1 A = I$.

If the transformations are applied simultaneously to I and the matrix B , the result is the inverse matrix A^{-1} , and X , where $AX = B$.

Note

It may be observed that this subroutine is identical to FORTRAN IV Library Routine AL **MTNV**, with the included feature, here, of variable dimensions for the arrays A and B . A double-precision counterpart to AL MINV has been prepared, namely AL DPMI.

References

Bodewig, E.: Matrix Calculus. North Holland Publishing Co., Amsterdam, 1959, pp. 182-183.

FORTRAN IV Library Routine AL **MTNV** (F1-1.0)

FORTRAN IV Subroutine AL DPMI

Identification

AL MTNV (MATINV), Matrix Inversion and Solution of Systems of Linear Equations

FORTTRAN IV

Ames Modification of SHARE Library Routine AN F402

Purpose

This program is used to solve the matrix equation $AX = B$, where A is a square coefficient matrix, and B is a matrix of constant vectors. The inverse matrix A^{-1} is obtained, and an option is provided to calculate the inverse matrix and omit the solution of the matrix equation. The determinant is also evaluated.

Usage

This subroutine is entered by use of the statement

```
CALL MATINV(A,N,B,M,DETERM)
```

where

A is the array that contains the coefficient matrix. The coefficients are to be stored columnwise in A, e.g., in the order $a_{11}, a_{21}, \dots, a_{n1}, a_{12}, a_{22}, \dots, a_{n2}$, etc.

N is the order of the matrix A ($N \leq 20$)

B is the array that contains the constant column vectors

M is the number of column vectors

DETERM is the location where the value of the determinant is to be placed

Upon return, the inverse matrix A^{-1} will be found in the array A, and the roots X will be found in the array B. If this subroutine is entered with $M = 0$, then the inverse matrix A^{-1} will be computed, but the computation of X will be omitted.

Restrictions

This subroutine has been compiled to solve systems of as many as 20 equations. The programmer must set up arrays as follows: A(20,N), B(20,M). If it is desired to solve systems of more than 20 equations, then MTNV can be recompiled with suitably revised DIMENSION and COMMON statements or the subroutine AL MINV may be used.

5/10/64

Restrictions (continued)

COMMON has been reserved in this subroutine by use of a BLOCK COMMON statement. This COMMON area has been labelled "ANE206", and so the latter symbol may not be employed by the programmer. The COMMON reserved here is compatible with that set aside in the subroutine AL LSQPOL, which uses AL MTNV internally. The programmer need not reserve any COMMON storage for use by MATINV.

Example

The following example is shown to illustrate the storage of coefficients in array A.

$$\begin{aligned} X + 3y + 6Z &= 10 \\ 2X + 4y + 10Z &= 20 \\ 6X + 9y + 7Z &= 13 \end{aligned}$$

$A(1,1) = 1.0$	$A(1,2) = 3.0$	$A(1,3) = 6.0$
$A(2,1) = 2.0$	$A(2,2) = 4.0$	$A(2,3) = 10.0$
$A(3,1) = 6.0$	$A(3,2) = 9.0$	$A(3,3) = 7.0$

Method

Jordan's method is used to reduce the matrix A to the identity matrix I, through a succession of elementary transformations, $T_n T_{n-1} \dots T_1 A = I$. If these transformations are applied simultaneously to I and the matrix B, the result is the inverse matrix A^{-1} and X, where $AX = B$.

Reference

Bodewig, E.: Matrix Calculus. North Holland Publishing Co., Amsterdam, 1959. pp. 182-183.

Identification

AL NDID Subroutine to Rewind, Backspace, and Place End-of-File
Marks on Magnetic Tape

FORTTRAN IV, MAP

Written by B. L. Meyer

Purpose

This subroutine is used to rewind, backspace, and place end-of-file marks on tapes. It may be used in conjunction with either FORTRAN or MAP input/output procedures, but it must be used for the latter.

Usage

There are four entries to this subroutine.
These are shown in the chart below

CALL REWND(N)	Rewind Tape N
CALL BACKRC(N)	Backspace tape N one physical record
CALL BACKFL(N)	Backspace tape N one file
CALL WREOF(N)	Write end-of-file mark on tape N

Identification

AL PAGE, Print Date, Page Number, and Title
 FORTRAN IV, MAP-coded
 Written by V. L. Sorensen

Purpose

The purpose of this subroutine is to allow the programmer to place the date and page number at the top of each sheet of printed output. An option is provided to print a title along with the date and page number.

Usage

There are three entries to this subroutine, which will be discussed below.

A.- CALL TITLE(nH...(title)..,I)

n is an integer constant ≤ 108

I is an option word

I = 0 page number is to be initialized

I = 1 page number is cumulative

This entry will cause ejection to the top of a new page. A fixed title, as given in the first item of the calling sequence, will be printed along with the date and page number.

B.- CALL VTITLE(V,J,I)

V is an array containing the desired title

J is the number of BCD words in the title (≤ 18)

I is as in the discussion of TITLE, above

This entry will accomplish the same results as does CALL TITLE, except that the title V is not necessarily fixed, but may be conveniently altered with the use of BCD input cards at execute time.

C.- CALL PAGE

If either of the two entries, TITLE or VTITLE, has been called then it is sufficient to use CALL PAGE for each new page. The title that is printed, along with the date and page number, is the one previously stored by the CALL TITLE or CALL VTITLE statement. The page incrementing is accomplished by simply placing the CALL PAGE statement in a line-count loop.

Identification

AL PALP, Draws Alphanumeric and Special Characters on EAI Dataplotter Plots Using the Pen
 FORTRAN IV, MAP-coded
 EAI Subroutine, with modifications by Melba Perniciaro

Purpose

This subroutine is used to draw any of 48 alphanumeric and special characters (including blank) to the desired size at a desired location in either the X or Y direction. Its primary use is the titling and annotation of plots prepared with the FORTRAN IV routines AL PLTW(PLOTWS) or AL MPLT.

Usage

The calling sequence for AL PALP is:

```
CALL PALP(ORIGX, ORIGY, SCALE, ARRAY, NUM, NTAPE, NXORY)
```

The symbols in this CALL statement are defined as follows:

ORIGX } ORIGY }	The horizontal and vertical coordinates, respectively, (measured in signed inches from the center of the page of plotting paper) of the lower left hand corner of the first character to be drawn.
SCALE	The height of the characters to be drawn (in inches).
ARRAY	An array containing the alphanumeric characters (including blanks) to be drawn, 6 characters/word, or a Hollerith string of the form nH..., where n*NUM.
NUM	The number of characters (including blanks) to be drawn.
NTAPE	The logical number of the plot tape.
NXORY	A code word controlling the direction the characters are drawn: If NXORY = +1, the characters are drawn in the X direction. If NXORY = -1, the characters are drawn in the Y direction.

Restrictions

The argument "ARRAY" in the calling sequence must be one of the following:

1. A Hollerith string of the form nH..., where n is the number of characters (including blanks) to be drawn.
 Example: CALL PALP (ORIGX,ORIGY,SCALE,
 23HTHIS IS A SAMPLE TITLE.,23,NTAPE,NXORY)
2. An array containing alphanumeric information in BCD form, 6 characters/word.
 The array can be filled from cards using the "A" format or with a DATA statement.

There are presently two EAI Dataplotters in use. For both plotters, the minimum character height is 0.15".

The plotter must be in the "line" mode to draw characters correctly. Thus, for "point" plots the programmer should write an end-of-file on the plot tape at the point where manual intervention is required to change the plotting mode.

Although the plotting paper is approximately 30 inches square, the actual area available for plotting is quite different. In the X direction, we must allow an inch on either side of the paper to keep the pen and printer arms from colliding. There is no restriction on the pen in the Y direction. However it is wise to allow for the addition of symbol printing to any plot, and for the printers we must allow 1-3/4" on either side for the width of the printing attachment. Thus, it is recommended that all plotting information, and thus all characters to be drawn by AL PALP, be confined to an area of 28 inches by 26½ inches.

Discussion

The 48 characters available include:

- 1) All alphabetic letters: A-Z
- 2) All numbers: 0-9
- 3) Special Characters: .,=/()\$+ -*, where "-" is either a minus sign or a dash.
- 4) A blank for spacing.

See the attached example for the actual shapes of the characters drawn by the plotter.

In general, the characters have a width to height ratio of 1.1.

There are 7 exceptions:

W	1.2
(,)	0.55
. 1 I	0.5

Thus, as a rough estimate, 10 characters, 1/4" high, to be drawn along the X-axis would take a space of 1/4" in the vertical direction and $10 \times 1/4 \times 1.1 = 11/4$ " in the horizontal direction. The same letters on the Y-axis would take 11/4" in the vertical direction and 1/4" in the horizontal.

The characters to be drawn by AL PALP will generally be fairly small, i.e., the distance between successive points is small. Because of these small displacements, the letters can be plotted rapidly, perhaps more so than the rest of the plot. Thus, if possible, all labels should be put in a separate file (designated as label information on the plot request card) so that the plotter speed can be changed.

Examples

(See attached sheets)

- I. The first five lines illustrate different size letters:
1", 1/2", 1/4", .2", .15". The following coding was used:

```
DATA ALPHA / 6HABCDEF /  
CALL PALP(0.0,ORIGY,SCALE,ALPHA,6,7,+1)
```

where ORIGY assumed the values 9.0, 8.0, 7.25, 6.5, 5.75
while SCALE was 1.0, 0.5, 0.25, 0.2, 0.15.

For the graph at the bottom of the page, AL PLOT was used for the curve and axes; AL SFAC produced the scale factors on the axes (using the symbol printing accessory of the EAI Dataplotter); and AL PALP labeled the plot. In this case, the characters to be drawn were read into arrays OMS (dimension 4), CYC (dimension 5), and CAP (dimension 2) from cards, using the "A" format. The calling sequences for AL PALP were:

```
CALL PALP (-0.5, 1.5, 0.15, OMS, 23, 7, -1)  
CALL PALP ( 0.5, 0.25, 0.15, CYC, 29, 7, +1)  
CALL PALP ( 4.0, 3.5, 0.15, CAP, 11, 7, +1)
```

The letters along the right side of the page illustrate the actual appearance of all the characters available. A string of Hollerith characters was used in the CALL statement:

```
CALL PALP (6.5, 1.5, 0.15, 48HABCD...()$-,48,7,-1)
```

- II. The second page of examples shows the labels to be used on a set of 18 plots.

Identification

AL PLOT, Routine for Entrance to AL PLTW(PLOTWS)

FORTRAN IV

Written by Melba Perniciaro

This is a dummy subroutine that calls AL PLTW(PLOTWS) by using the statement CALL PLOT (). Writeups of AL PLOT are available in the CAB library files.

```

    DIMENSION STALOC(2,20), IDATE(2,20), ITIME(20), ARRAY(5), ORIG(20,2),
1  WORK(5), IWORK(5), FMT1(4), FMT2(1)
    EQUIVALENCE (WORK(1), IWORK(1))
    DATA(FMT1(10), I0=1,4)/24H(2A6,2H, ,A6,A2,2H , ,I4)/,FMT2(1)/
1  5H(5A6)/

```

```

    :
    KOUNT=0
    DO 101 I=1,NOSTA
    WORK(1)=STALOC(1,I)
    WORK(2)=STALOC(2,I)
    DO 101 J=1,NODATE
    IWORK(3)=IDATE(1,J)
    IWORK(4)=IDATE(2,J)
    DO 101 K=1,NOTIME
    IWORK(5)=ITIME(K)
    CALL CVRT (WORK,5,FMT1,ARRAY,5,FMT2)
    KOUNT=KOUNT+1
    CALL PALP (ORIG(KOUNT,1),ORIG(KOUNT,2),0.15,ARRAY,28,7,+1)
101 CONTINUE

```

Note that the numerical information in the labels was generated within the program. It was converted to the proper form by using the FORTRAN IV subroutine AL CVRT.

References

1. FORTRAN IV Library Routines AL PLTW and AL PLOT.
2. FORTRAN IV Routine AL MPLT.
3. FORTRAN IV Routine AL SFAC.
4. FORTRAN IV Routine AL CVRT.

A B C D E F

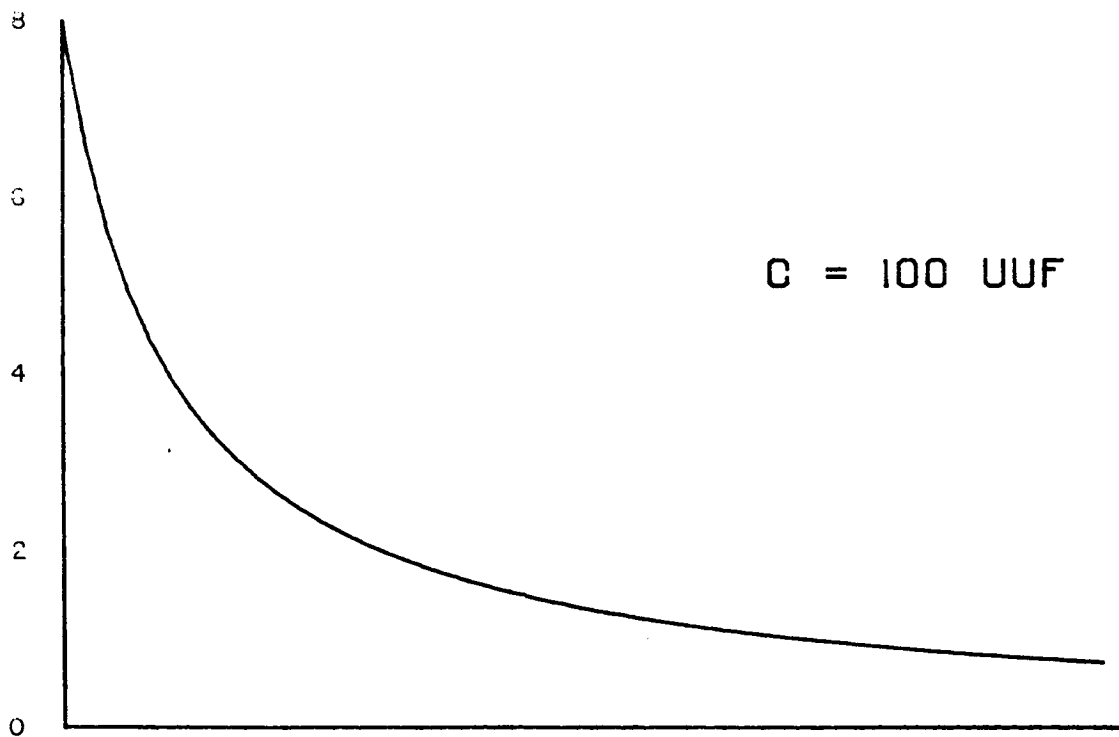
A B C D E F

A B C D E F

A B C D E F

A B C D E F

REACTANCE - OHMS X 1000



ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789=-*./,()\$-

FREQUENCY - KILOCYCLES X 1000

JOHANNESBURG, 12/01/64, 930
JOHANNESBURG, 12/01/64, 1530
JOHANNESBURG, 04/16/66, 930
JOHANNESBURG, 04/16/66, 1530
JOHANNESBURG, 08/31/68, 930
JOHANNESBURG, 08/31/68, 1530
PUERTO RICO , 12/01/64, 930
PUERTO RICO , 12/01/64, 1530
PUERTO RICO , 04/16/66, 930
PUERTO RICO , 04/16/66, 1530
PUERTO RICO , 08/31/68, 930
PUERTO RICO , 08/31/68, 1530
GRAND BAHAMA, 12/01/64, 930
GRAND BAHAMA, 12/01/64, 1530
GRAND BAHAMA, 04/16/66, 930
GRAND BAHAMA, 04/16/66, 1530
GRAND BAHAMA, 08/31/68, 930
GRAND BAHAMA, 08/31/68, 1530

November 1, 1965

Identification

AL PLTW(PLOTWS), Plots the Array Y Versus the Array X on the EAI Dataplotter
 Using Either the Pen or the Symbol Printer
 FORTRAN IV, MAP coded
 Written by Melba Perniciaro, April, 1965

Purpose

This subroutine is used to plot an array of ordinates Y versus an array of abscissas X on an EAI Dataplotter. The locations of these points on the plotting paper are determined by the origin and scaling information supplied by the programmer. Either the pen or the symbol printer may be used. (If the pen is used, either lines or points may be specified on the plot request card).

Usage

The EAI Dataplotter plots points, lines, or symbols on a sheet of paper that is approximately 30 inches square. (See the heading "Restrictions" for a discussion of the actual area available for plotting). The origin of a given plot is referred to a fixed system of coordinates whose origin is at the center of this 30"x 30" page. (See the sketch at the end of this summary).

The calling sequence for AL PLOTWS is:

CALL PLOTWS (ORIGX, ORIGY, SCALEX, SCALEY, X, Y, NUM, NTAPE, NSYM, NER)

The symbols in the CALL statement are defined as follows:

ORIGX } The horizontal and vertical coordinates, respectively of the
 ORIGY } origin of the plot, measured in signed inches from the center
 of the page of plotting paper.

SCALEX } The number of units per inch in the X and Y (horizontal and
 SCALEY } vertical) directions, respectively.

X } The arrays containing the X and Y values, respectively, of the
 Y } points to be plotted. More generally, X and Y are the locations
 of the first X and Y values, respectively, to be plotted, and
 the other values are stored sequentially following the first.

NUM The number of points to be plotted.

NTAPE The logical number of the tape on which the plotter information
 is to be written

NSYM A code word controlling the pen and printer:

If NSYM is a negative non-zero integer, the pen is selected, i.e.,
 no symbols are to be printed.

If NSYM contains a character of Hollerith information, e.g., NSYM = 1HA,
 the appropriate symbol will be printed (see Table I).

NER an Error code:

If NER = 0, no error has occurred.

If NER = 1, NSYM does not correspond to any symbol in Table I.

Restrictions

Although the plotting paper is approximately 30 inches square, the actual area available for plotting is quite different. In the X direction, we must allow an inch on either side to keep the pen and printer arms from colliding. Thus, at all times, $|X|$ must be ≤ 14.0 ". In the Y direction, there are restrictions on the printers, but not on the pen. This allowance is 1-3/4 inches on either side. Since it is wise to allow for the addition of symbol printing to any plot, it is recommended that, at all times, $|Y|$ be ≤ 13.25 ". Thus, plots should be confined to an area of 28 inches by 26½ inches.

For symbol plotting the argument "NSYM" in the calling sequence must have one of the following forms:

1. A Hollerith character written directly in the calling sequence.
Example: CALL PLOTWS (ORIGX, ..., NTAPE, 1HA, NER)
In this case, the symbol desired is the letter "A".
2. A variable containing one character of alphanumeric information in BCD form. The BCD code for the desired character must be in the left-most portion of the word, and the remaining positions must contain the BCD equivalent of blanks. NSYM can be filled from cards using a format of "A1" or with a DATA statement.

Discussion

The plotters are run with the arm select switch at the "auto" setting. Every time an entry is made to PLOTWS, the proper arm select command is given - pen or printer. A modification to the plotters allows redundant arm select commands to be ignored. (This is necessary since otherwise each arm select sends the plotter to the "stand-by" position, i.e., both arms move back to the side of the plotting table).

If symbols are to be printed on a line plot, then PLOTWS must be called twice. First, the symbols should be printed, using the appropriate setting for NSYM. Then, the lines should be plotted by setting NSYM to a negative number. The lines should be plotted last to avoid smearing the ink when the symbols are printed. Notice that, unless symbols are to be printed at every point of the line plot, the arrays X and Y are not the same for the two entries to PLOTWS.

When plotting with the pen, the plotting speed must be set manually. It is inversely proportional to the "maximum point displacement", i.e., maximum distance between adjacent points. Once this speed is selected, via the "Maximum Point Displacement" control on the plotter console, manual interruption of the plotting is required to change it. Now as an example, consider the plotting of a curve (such that the maximum distance between adjacent points is approximately 1/4") and a set of axes for this curve (where the maximum distance between adjacent points is perhaps 2"). First the Dataplotter operator must run through the plot with the pen up to determine how fast it can be plotted. Then, the entire plot must be done very slowly because the maximum displacement is 2". Thus, the following recommendations are made:

1. If possible, estimate the maximum point displacement for each file of plot information on the plot request card.
2. If you have many points which are close together and a few which are far apart, use separate files for the two sets of data. In our example, if the data for the curve and the axes had been in separate files, the former could then have been plotted with maximum efficiency, using a setting of 1/4".

PLOTWS may be called repeatedly if more than one set of data is to be placed on a sheet. After the last CALL PLOTWS statement and at any point in the plotting procedure where manual intervention is required of the Dataplotter operator (e.g., change of plotter paper, ink color, line or point control, or maximum point of displacement control), the programmer must write an end-of-file on the plot tape by use of the FORTRAN statement.

END FILE NTAPE

In addition it is desirable that an end-of-file be written at the beginning of each plot tape. In some applications it is desirable to plot with one or both axes reversed, i.e., rotated 180 degrees. This may be done with PLOTWS by making the appropriate scale factor a negative number.

Special Note




Attention is called to the fact that the programs AL PLOT and AL PSCA are now subroutines which convert the CALL PLOT and the CALL PSCA statements to the statements CALL PLOTWS, and CALL SFAC with appropriate alteration of the calling sequences.

Other FORTRAN IV Plotting Routines

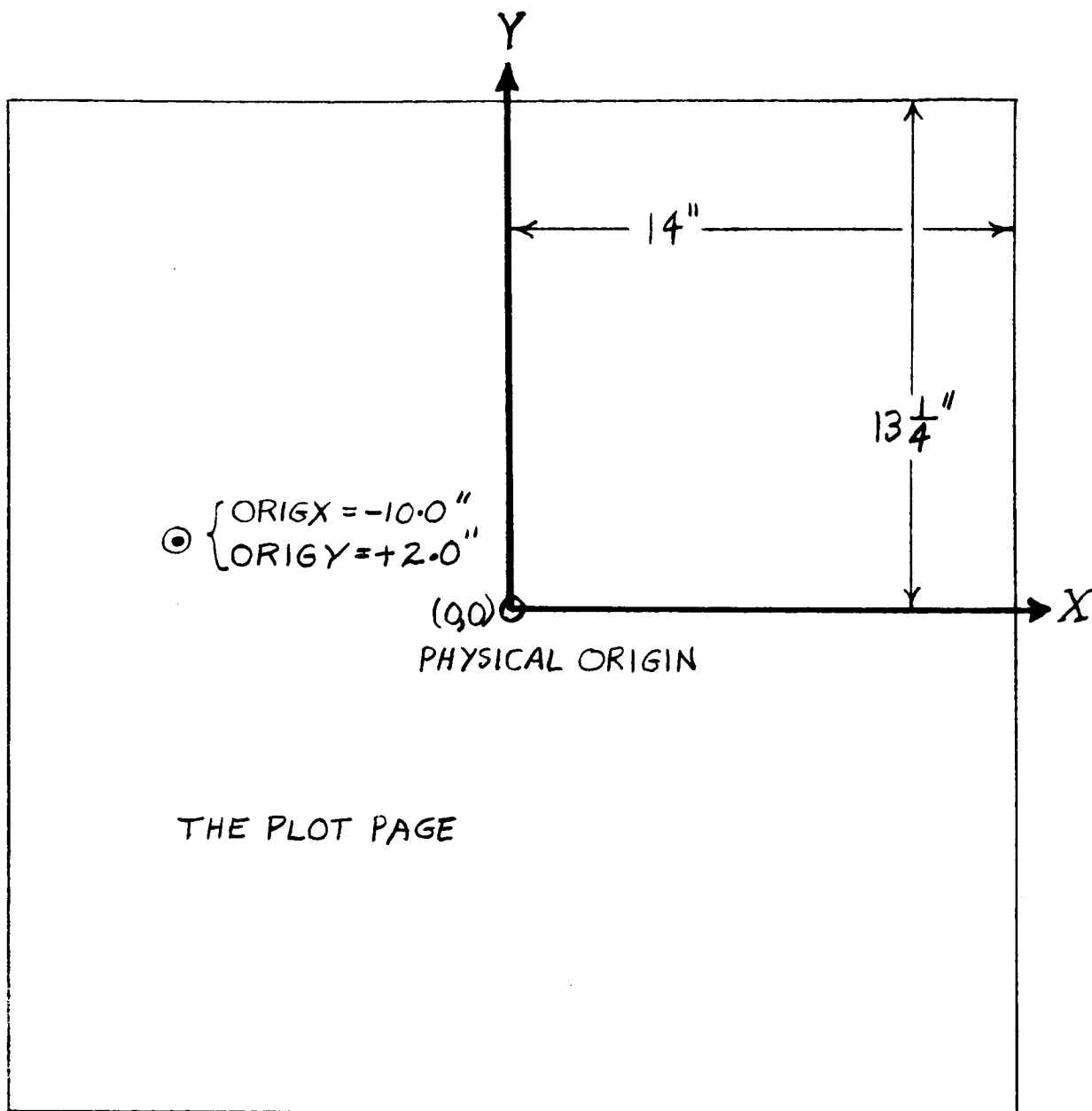
- AL SFAC, Prints X and Y scale factors of any magnitude, positive or negative, in any of 3 modes - integer, decimal, or decimal with exponent. SFAC can also print the value of a single variable anywhere on the plotting page.
- AL PALP, Draws alphanumeric and special characters for titling plots using the pen.
- AL ALPP, Prints alphanumeric and special characters for titling plots.
- AL MPLT, Plots data with automatic scaling and placement of origin so that the maximum plotting area is 12" x 12". Both linear and logarithmic scales are available. Special attention has been given to minimizing plotting time while maintaining the best possible accuracy.
- AL SCAL, Computes a scale factor, on the basis of a given axis length and the maximum and minimum values to be plotted.

Symbols Available on the Printer Arms

TABLE I

Hollerith Character (NSYM)	Symbol
0 - 9	0 - 9
A - Z	A - Z
+	+
- (minus)	- (minus)
.	.
,	,
((
))
*	*
/	/
=	=
- (dash)	
\$	
blank	

THE PLOT PAGE



THE PLOT PAGE

$$\odot \begin{cases} \text{ORIGX} = -1.0'' \\ \text{ORIGY} = -20.0'' \end{cases}$$

Identification

AL POLY(POLYEL), Evaluation of a Polynomial
 FORTRAN IV, MAP-coded
 Taken from SHARE Library Routine AN E206 (AL LSQP)

Purpose

This subroutine is used to evaluate the polynomial

$$y = b_1 + b_2x + b_3x^2 + b_4x^3 + \dots + b_Mx^{M-1}$$

where the coefficients b_1, b_2, \dots, b_M , and the argument x , are given.

Usage

This subroutine is entered by use of the statement

$Y = \text{POLYEL}(X, M, B)$

where

X is the value of the dependent variable

M is the number of coefficients in the polynomial

B is an array of dimension at least M that contains the coefficients of the polynomial, e.g.,

$B(1) = b_1$

$B(2) = b_2$

.

.

.

$B(M) = b_M$

Y is the value of the polynomial for the given value of X .

Comment

This subroutine is used internally by the subroutine AL LSQP(LSQPOL), but it is available ~~for~~ general use with no restrictions.

Method

The polynomial

$$y = b_1 + b_2x + b_3x^2 + \dots + b_Mx^{M-1}$$

is evaluated by use of the algorithm

$$y = (\dots((xb_M + b_{M-1})x + b_{M-2})x + \dots)x + b_1$$

Identification

AL PSCA(PSCALE)
FORTRAN IV

Purpose

This subroutine generates the calling statement for AL SFAC from the calling statement for AL PSCALE. Whenever possible SFAC should be used directly. It is a more general routine. The full writeup for AL PSCA will be found in the CAB library files.

Identification

AL RDM, Random Number Generator

FORTTRAN IV, MAP-coded

Ames modification of SHARE Library Routines GC 0008, GC 0010, GC 0011

Purpose

This subroutine is used to generate random numbers that are uniformly distributed on the unit interval. Options exist to save the place in the sequence of random numbers for a later restart.

Usage

The arithmetic statement

$$R = \text{RDM}(\text{DUMMY})$$

is used. The desired random number will be in the cell designated R. The symbol DUMMY need not be defined in the calling program, and it will not be altered by the subroutine. If it is desired to save the place in the random sequence, so that upon restarting the program the random numbers obtained will be a continuation of the original sequence, the statement

$$\text{CALL RDMOUT}(\text{OCT})$$

is used. The cell OCT contains an octal number which should be read out using O-format. This octal number is then loaded into the program prior to the restart, again using the O-format. To re-initialize the random number generator, the statement

$$\text{CALL RDMIN}(\text{OCT})$$

is used, where OCT is the previously saved octal number.

Method

The sequence of computations is

$$R_{i+1} = (2^7 + 1)R_i + 311715164025_8$$

The resulting fixed point number is converted to floating point form. The method is that of Rotenberg (ref. 1). The octal constant in the above relation was chosen since it is odd and approximates $[(.5 + \sqrt{3}/6)2^{35}]_{10}$ which is shown by Coveyou (ref. 2) to be that constant which causes the least serial correlation.

References

- 1.- Rotenberg, A.: A New Pseudo-Random Number Generator. Jour. of the A.C.M., vol. 7, no. 1, Jan. 1960.
- 2.- Coveyou, R. R.: Serial Correlation in the Generation of Random Numbers. Jour. of the A.C.M., vol. 7, no. 1, Jan. 1960.

Identification

AL REWN(REWUN), Rewind and Unload a Magnetic Tape
FORTRAN IV, MAP-coded
Written by V. L. Sorensen

Purpose

The use of this subroutine allows the programmer to rewind and unload a logical magnetic tape.

Usage

A logical tape is rewound and unloaded by use of the statement

CALL REWUN(N)

where N is the logical tape number

Restrictions

The tape to be rewound and unloaded must be one that has been mounted specifically for the job at hand. It may not be used for the standard input and output tapes 5 and 6, and should not be used with a tape that has been written by the plotting subroutine, AL PLTW.

Identification

AL RNDM, Random Number Generator

See AL BARN, AL RNDM, Ames FORTRAN IV Library subroutines.

Identification

AL ROOT, Calculate the Root, Within Specified Bounds, of a Function
 FORTRAN IV
 Written by V. L. Sorensen

Purpose

This subroutine is used to calculate a root of the function
 $f(x) = 0$

Usage

This subroutine is entered by use of the statement

```
CALL ROOT(F,X,G,BL,BU,E1,E2,I)
```

where

- F is the name of a function subprogram with a single argument, X, that is used to evaluate $f(x)$. It must be named in an EXTERNAL statement.
- X is the root and the argument of the function subprogram F.
- G is an initial guess, supplied by the programmer.
- BL is the lower bound on X.
- BU is the upper bound on X.
- E1 } are numbers that specify the degree of precision expected
 E2 } of the root determination. See subsection D under Method
- I is an error code:
 - I = 1 normal return
 - I = 2 G is outside of the specified limits
 - I = 3 BU is smaller than or equal to BL
 - I = 4 two successive iterations produce the identical function value, but the convergence criteria are not satisfied
 - I = 5 a root cannot be found within the given bounds

Usage (continued)

An example is presented to illustrate the usage as discussed above.

```

EXTERNAL FUN
BL = 0.0
BU = 10.0
G = 5.0
E1 = 1.0E-6
E2 = 1.0E-6
CALL ROOT(FUN,X,G,BL,BU,E1,E2,I)
GO TO(1, 2,2,2,2), I
1 -
-
-
CALL EXIT
2 CALL ERROR(2,0)
END

```

The function subprogram FUN might be:

```

FUNCTION FUN(X)
FUN = X**3 + X**2 - 2.0
RETURN
END

```

Method

The root, if it exists, is determined by an iterative technique based on the Newton-Raphson method. The iteration relation is

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

and the derivative $f'(x_n)$ is approximated by the linear relation

$$f'(x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

The indexes $n+1$, n , $n-1$ refer to the new value and two previous values respectively. Four cases can now be distinguished.

Method (continued)

A. Initially, x_n and x_{n-1} are taken to be

$$\begin{aligned}x_n &= G1 \\x_{n-1} &= 0.1(G1 - BL) + BL\end{aligned}$$

where

$$\begin{aligned}G1 &= G \text{ if } BL \leq G \leq BU \\&= 0.5(BL + BU) \text{ if } G = BL \text{ or if } G = BU\end{aligned}$$

B. Subsequently, if x_{n+1} falls outside of the specified upper and lower bounds, new starting values are inserted for x_n and x_{n-1} . Five such restarts are provided, and if none lead to the desired root, no further attempt is made. The five pairs of starting values are:

- (1) $G1, BU - 0.1(BU - G1)$
- (2) $BL + 0.1(G1 - BL), 0.5(BL + G1)$
- (3) $0.5(BL + G1), G1$
- (4) $G1, 0.5(BU + G1)$
- (5) $0.5(BU + G1), BU - 0.1(BU + G1)$

The symbol $G1$ has been defined in A above.

- C. If the root has not been bracketed by two values of x_1 within the specified bounds such that one gives a positive value for $f(x)$ and the other a negative value for $f(x)$, then the values taken for x_n and x_{n-1} are the two produced by the immediately preceding iterations.
- D. If the root has been bracketed as described in C, then x_n and x_{n-1} refer to the most recent iterations for which $f(x_n)$ was positive and $f(x_{n-1})$ negative.

The iteration process is continued until either of two convergence criteria is satisfied. These criteria are

$$(1) \quad |f(x_{n+1})| \leq E1$$

and

$$(2) \quad \left| \frac{x_{n+1} - x_n}{x_{n+1}} \right| \leq E2 \quad \text{if } x_{n+1} \neq 0$$

Method (continued)

If the root has been bracketed, then test (2) is repeated with x_n replaced by x_{n-1} .

Identification

AL ROP, Roots of a Real or Complex Polynomial
 FORTRAN IV, MAP-coded
 Ames Modification of SHARE Library Routine GL ROP1

Purpose

This subroutine is used to compute all of the roots, real and complex, of the polynomial

$$a_0x^N + a_1x^{N-1} + \dots + a_N = 0$$

where the coefficients a_0, a_1, \dots, a_N may be real or complex.

Usage

This subroutine is entered by use of the statement

CALL ROP(N,P,R)

where

N is the degree of the polynomial

P is the array of dimension at least $2N + 2$ where the coefficients a_0, \dots, a_N are stored

R is the array of dimension at least $2N$ where the polynomial roots are to be stored.

The coefficients of the polynomial are stored as follows:

Re a_0 P(2N + 2)

Im a_0 P(2N + 1)

· ·

· ·

· ·

Re a_N P(2)

Im a_N P(1)

Upon return from the subroutine, the roots will be found in the array R as follows:

Usage (continued)

Re x_1	R(2N)
Im x_1	R(2N - 1)
.	.
.	.
.	.
Re x_N	R(2)
Im x_N	R(1)

Caution

The arrays P and R are not arranged in a manner that is directly compatible with the use of FORTRAN IV complex arithmetic.

Method

An initial approximation is made to a root, and the order of the polynomial is then reduced by synthetic division. Each root is refined using the original polynomial.

Identification

AL SCAL, Scale Factor for Plotting Purposes

FORTRAN IV

Written by James A. Jeske

Purpose

This subroutine is used to compute a scale factor, on the basis of a given axis length in inches, and minimum and maximum values to be plotted on that axis. The computed scale factor will be one of the following:

$$\begin{array}{l} 1 * 10^N \\ 2 * 10^N \\ 4 * 10^N \\ 5 * 10^N \\ 8 * 10^N \end{array}$$

where N is a positive or negative integer. The scale factor will be determined such that the absolute value of the maximum value to be plotted will not exceed the specified curve length by more than one inch.

Usage

The subroutine is entered by use of the call statement

```
CALL SCALE(VMAX,VMIN,PS,SF)
```

where

VMAX	is the maximum value to be plotted
VMIN	is the minimum value to be plotted
PS	is the length, in inches, of the axis that corresponds to VMAX and VMIN
SF	is the scale factor, units per inch, as computed by the subroutine

Example

Suppose that the following numbers apply in a certain problem:

XMAX = 14.6 sec	YMAX = 1.2 lbs
SMIN = 2.3 sec	YMIN = 18.3 lbs
SIZX = 10.0 in	SIZY = 8.0 in

The calling statements

```
CALL SCALE (XMAX, XMIN, SIZX, SCALEX)
CALL SCALE (YMAX, YMIN, SIZY, SCALEY)
```

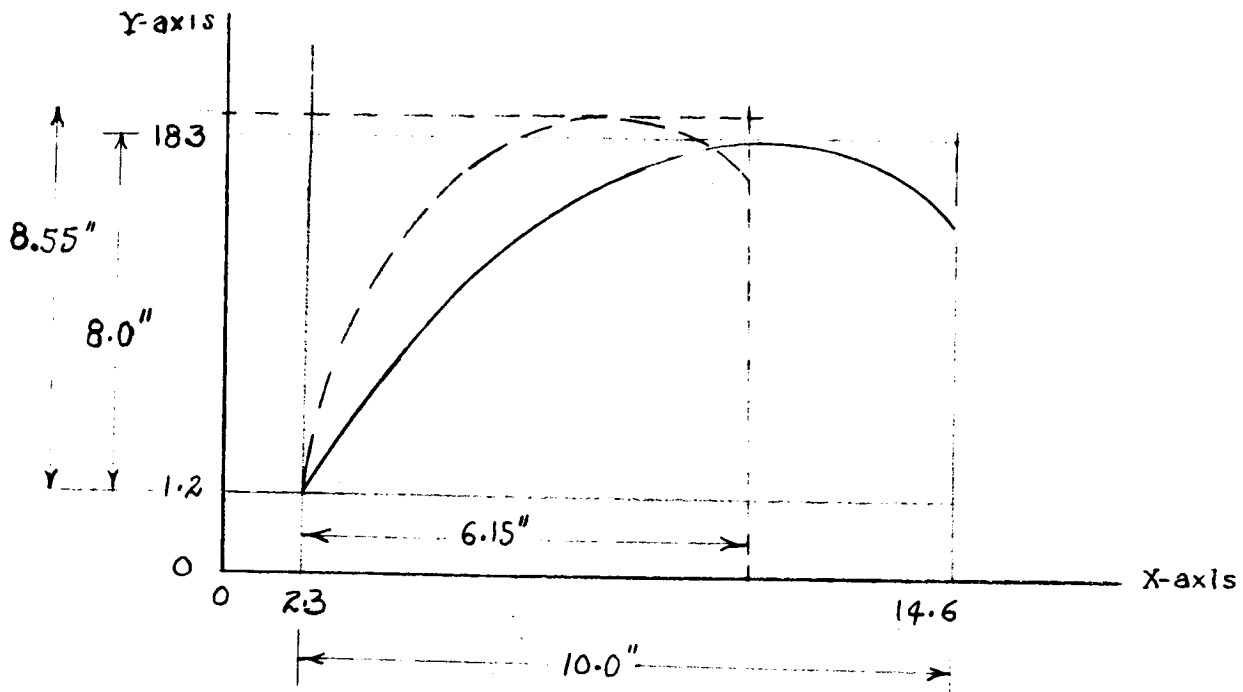
would compute scale factors

SCALEX = 2.0 sec/in

SCALEY = 2.0 lbs/in

The scale factor SCALEX has been picked so that the length on the X-axis that corresponds to the time interval XMAX-XMIN, here 12.3 sec, does not exceed the specified SIZX by more than one inch. If a scale factor of 1.0 had been chosen, the specified value of 10 inches would have been exceeded by 2.3 inches, and so the next best value, namely 2.0 has been computed. The value calculated for SCALEY, namely 2.0, allows the maximum value to plot 0.55 inches beyond the specified 8.0 inch limit, which, is acceptable. The sketch below illustrates the example presented here.

SKETCH



The solid line is the specified curve. However, with the scale factors that were computed, the broken curve will be the actual plotted curve.

Identification

AL SFAC, Prints Scale Factors or the Value of a Single Variable on EAI Dataplotter Plots
 FORTRAN IV, MAP coded

Written by Melba Perniciaro

Purpose

This subroutine has 2 functions:

1. To print X and/or Y scale factors at one inch intervals along the axis or axes of a plot. The plot itself is prepared using the FORTRAN IV routines AL PLOTWS or AL MPLT.
2. To print the value of a single variable anywhere on the plotting page.

This is especially useful in preparing title information for plots.

For both functions, positive and negative numbers of any size can be printed in a choice of three modes: integer, decimal, or decimal with exponent.

Usage

The calling sequence for AL SFAC is:

```
CALL SFAC (ORIGX, ORIGY, SCALEX, SCALEY, NUMX, NUMY, VORIGX, VORIGY,
           NTAPE, MODE)
```

The symbols in the CALL statement are defined as follows:

I. To print scale factors:

ORIGX } ORIGY }	The horizontal and vertical coordinates, respectively, (measured in signed inches from the center of the page of plotting paper) of the origin of the axis or axes along which printing is to be done.
SCALEX } SCALEY }	The number of units per inch in the X and Y (horizontal and vertical) directions, respectively.
NUMX } NUMY }	The number of scale factors to be printed on the X and Y axes, respectively.
VORIGX } VORIGY }	The first values to be printed on the X and Y axes, respectively.
NTAPE	The logical number of the plot tape.
MODE	A parameter controlling the format of the numbers printed (see the heading "Discussion"):
	If MODE = 1, use the integer mode.
	If MODE = 2, use the decimal mode.
	If MODE = 3, use the decimal mode with exponent.

If NUMX = 0, no scales will be printed on the X axis, and a similar statement applies to NUMY. In either case, the corresponding scale factor (SCALEX or SCALEY) and initial value (VORIGX or VORIGY) are not used and may have any value.

II. To print a single variable:

ORIGX } ORIGY }	The horizontal and vertical coordinates, respectively, (measured in signed inches from the center of the page of plotting paper) of the starting point for printing the value of the variable.
SCALEX	Set to 0.
SCALEY	This is not used and may have any value.
NUMX	Set to -1.
NUMY	Set to 0.

VORIGX The value of the variable to be printed.
 VORIGY This is not used and may have any value.
 NTAPE The logical number of the plot tape.
 MODE See the description in I. above.

If desired, the roles of NUMX and NUMY may be interchanged. (Interchange also the roles of SCALEX and SCALEY and VORIGX and VORIGY). The only difference in the results is the position of the exponent in mode 3. For NUMX = -1, it appears to the right of the number; for NUMY = -1, it is above the number. (See the heading "Discussion").

Restrictions

Regardless of the mode being used for printing, i.e., integer, decimal, etc., the variables in the calling sequence always have the same mode:

ORIGX, ORIGY, SCALEX, SCALEY, VORIGX and VORIGY are floating point variables, while NUMX, NUMY, NTAPE, and MODE are integers.

The subroutine SFAC will perform all necessary rounding and conversion procedures.

Although the plotting paper is approximately 30 inches square, the actual area available for plotting is quite different. In the X direction, we must allow an inch on either side of the paper to keep the pen and printer arms from colliding. Thus, at all times, $|X|$ must be ≤ 14.0 ". In the Y direction, we must allow 1-3/4" on either side for the width of the 48 character printing attachment. Thus, $|Y|$ must be ≤ 13.25 ". This means that all plotting information must be contained in an area 28 inches by 26½ inches.

Discussion

The 3 modes available for the numbers to be printed are:

1. Integer - A maximum of 6 digits and a sign (if negative) will be printed.
2. Decimal - A maximum of 5 digits, a decimal point, and a sign (if negative) will be printed.
3. Decimal with exponent - Each number is printed in the decimal mode above. One-fourth inch to the right of or above the last scale on the X and Y axes, respectively, an exponent is printed. The exponent consists of a sign (if negative) and a maximum of 2 digits, preceded by a symbol to indicate that what follows is an exponent.

In all cases, only significant digits are printed. Extra zeros to the left or right of the number are suppressed.

For scale factors on the X axis, the numbers are printed so that they are centered about (ORIGX, ORIGY - 0.25), (ORIGX + 1.0, ORIGY - 0.25), etc. On the Y axis, the scales are centered about (ORIGX - 0.5, ORIGY), (ORIGX - 0.5, ORIGY + 1.0), etc. For single variables, the center of the number, excluding any exponent, is at (ORIGX + 0.3, ORIGY).

Within any particular scale factor or exponent, corresponding points on adjacent characters are 0.1" apart.

Notice that when printing scale factors, the values specified for ORIGX and ORIGY are the origin of the axis or axes along which printing is to be done. This is usually not the same as the origin of the plot, i.e., the location of (0.0, 0.0).

For neatness, it is recommended that scale factors be printed around the outside perimeter of the plot.

To achieve maximum efficiency, this routine is written to use the "select and print" mode of an EAI Dataplotter equipped with a 48 character symbol printer. In this mode there is virtually no delay for changing the character to be printed since this is done as the plotter moves to the point where the new symbol is to be written.

Examples

The following examples illustrate the usage of AL SFAC. All numbers on the page were printed using SFAC, while AL PALP did the lettering. Notice the use of the single variable function of SFAC to print the numbers for the headings: "Example 5", "Example 6", etc.

Example: CALL PALP (1.5, 1.5, 0.15, 7HEXAMPLE, 7, 7, +1)
CALL SFAC (2.5, 1.55, 0.0, 0.0, -1, 0, 7.0, 0.0, 7, 1)

The following calling sequences were used for the examples:

Examples 1 - 4 illustrate the single variable function of SFAC

1. CALL SFAC (7.0, 7.0, 0.0, 0.0, -1, 0, -123456.0, 0.0, 7, 1)
2. CALL SFAC (7.0, 8.0, 0.0, 0.0, 0, -1, 0.0, 12.345, 7, 2)
3. CALL SFAC (7.0, 9.0, 0.0, 0.0, -1, 0, 12.345E-17, 0.0, 7, 3)
4. CALL SFAC (7.0, 10.0, 0.0, 0.0, 0, -1, 0.0, -0.9999E32, 7, 3)

Examples 5 - 7 illustrate the scale factor function of SFAC. In example 5, we called the subroutine twice to get decimal numbers on the X axis and integers on the Y axis. In example 6, we used a negative scale factor on the X axis. (This can be done on either axis). Example 7 indicates the position of the exponents in mode 3 and also the variety of numbers that SFAC can handle.

5. CALL SFAC (5.0, 5.0, 0.11111, 3333.0, 4, 0, 0.0, -10000.0, 7, 2)
CALL SFAC (5.0, 5.0, 0.11111, 3333.0, 0, 6, 0.0, -10000.0, 7, 1)
6. CALL SFAC (3.0, 3.0, -0.1, 0.1, 6, 8, 0.3, -7.2, 7, 2)
7. CALL SFAC (1.0, 1.0, 1.0E34, 5.0E-34, 7, 10, 0.0, -28.0E-34, 7, 3)

References

Fortran IV Library Routine AL PLTW.
Fortran IV Routine AL PALP.

E-34

E 31

17.

-6.5

6565

4

-9.999

12.

-6.6

3332

3

1.2345 E-16

7.

-6.7

-1

2

12.345

2.

-6.8

-3334

1

-123456

-3.

-6.9

-6667

EXAMPLE 5

-8.

-7.

-10000

0.

.11111

.22222

.33333

-13.

-7.1

EXAMPLE 6

-19.

-7.2

.3

.2

.1

0.

-.1

-.2

-23.

EXAMPLE 7

-28.

0.

1.

2.

3.

4.

5.

6. E 34

Identification

AL SIMP, Integration by Simpson's Rule

FORTRAN IV, MAP-coded

Ames Modification of SHARE Library Routine CL INT4

Purpose

This subroutine is used to compute the integral of a function, tabulated at equal or unequal intervals.

Usage

The calling statement is

```
CALL SIMP(R, X, Y, N, I)
```

where

R is the result of the integration
 X is the name of the array that contains the independent variable
 Y is the name of the array that contains the dependent variable.
 N is the number of values in the arrays X and Y
 I is an error code
 I = 1 no error, normal return
 = 2 N=0 or N=1
 = 3 overflow or underflow has occurred
 = 4 the array of X-values is not monotonic

Restriction

The independent variable must increase or decrease monotonically, and N must be two or greater.

Method

The integral

$$R = \int_{X_1}^{X_N} Y \, dx$$

is evaluated by fitting parabolas to successive intervals and integrating over these intervals. The dependent and independent variables are given in tabular form, and the values of the independent variable need not be equally spaced. If N is odd, then the interval (X_1, X_N) is divided into $(N-1)/2$ intervals $(X_1, X_3), (X_3, X_5), \dots, (X_{N-2}, X_N)$. A parabola is fitted through the three points of each interval. The integral is then evaluated for each interval, and the results are summed to produce the integral over the entire interval (X_1, X_N) . If N is even, then a parabola is fitted through the three points X_1, X_2, X_3 , as above, and the integral is then evaluated in the interval (X_1, X_2) . The remaining points in the interval (X_2, X_N) are treated as in the case of odd N. The latter results are added to the integral over the interval (X_1, X_2) to give the integral over the interval (X_1, X_N) .

The formula

$$Y = Y_1 + (X - X_1) \left(\frac{Y_1 - Y_2}{X_1 - X_2} \right) + \frac{(X - X_1)(X - X_2)}{X_1 - X_3} \left[\left(\frac{Y_1 - Y_2}{X_1 - X_2} \right) - \left(\frac{Y_2 - Y_3}{X_2 - X_3} \right) \right]$$

is used for the parabolic fit to three adjacent points.

Identification

AL SINH-COSH, Hyperbolic Sine and Cosine Functions
FORTRAN IV, MAP-coded
Ames Adaptation of Similar Program in the ABC Compiler

Purpose

This subroutine is used to compute the functions $Y = \sinh(X)$ and $Y = \cosh(X)$, where X and Y are single-precision floating-point numbers.

Usage

This subroutine is entered by use of the statements

$Y = \sinh(X)$

and

$Y = \cosh(X)$

Error Condition

If this subroutine is entered with a value for X greater in absolute magnitude than 87.3, the value returned for the function is the largest machine number, e.g., 0.1701412E39.

Accuracy

The results from this subroutine are accurate to seven significant figures.

Timing

The average computing times are 393 μ s and 382 μ s for SINH and COSH, respectively.

Identification

AL TAIN(TAINT), Table Look-up and Interpolation
 FORTRAN IV, MAP-coded
 Written by V. L. Sorensen

Purpose

This subroutine is used to evaluate one or more functions of an argument X, when the functions and the argument are given in tabular form.

Usage

The CALL statement for this subroutine is

```
CALL TAIN (XTAB, FTAB, X, FX, N, K, NER, DMON, GTAB, GX, HTAB, HX, ...)
```

where

XTAB is the name of the array that contains the values of the argument of the functions, which must increase or decrease monotonically.

FTAB
 GTAB
 HTAB
 .
 .
 .

} are the names of arrays that contain the values of the functions of the argument in XTAB.

X is the value of the argument for which function values are sought.

FX
 GX
 HX
 .
 .
 .

} are the values of the functions as obtained by the subroutine.

N is the number of tabulated values, and so the arrays XTAB, FTAB, ..., are of dimension at least N; $N \geq K$.

K is the order of interpolation; $K \leq 9$.

Usage (continued)

NER is an error code:

NER = 1 normal return
 NER = 2 $K > 9$ or $N \leq K$
 NER = 3 adjacent values within the array XTAB
 are equal.

DMON is a variable that must be set to zero initially, and whenever XTAB is altered. Each XTAB within a given program should be assigned a unique DMON symbol. The contents of DMON are altered by the subroutine.

Restrictions

The table XTAB must be monotonic, and adjacent values may not be equal. The order K must be smaller than or equal to 9, and N must be greater than K.

Method

A binary search is used to find the region of the table where the desired result is to be found. Aitken's method of interpolation is then employed to determine the precise functional values corresponding to the given value of the independent variable. Aitken's method is a computational scheme for evaluating the Lagrangian interpolation polynomial without having to compute polynomial coefficients. The development of the Lagrangian interpolation method will be omitted here, as details may be found in many standard texts (i.e., see reference 1). The Aitken method (see reference 2) consists in the repetitive use of the formula

$$Y_{m+n}^n = \frac{Y_{n-1}^{n-1}(X_{m+n} - X) - Y_{m+n}^{n-1}(X_{n-1} - X)}{X_{m+n} - X_{n-1}}$$

The superscripts and subscripts are indices, and never exponents, in this formula. For a given value of the order K, the above formula is applied for n in the range 1, 2, ..., K, and for each n, the values for m are taken in the order 0, 1, ..., K-1. The restriction $n+m \geq K$ applies. Thus for linear interpolation (K=1), this process is carried out with a single application of the above general formula, e.g.,

Method (continued)

$$Y_1^1 = \frac{Y_0^0(X_1 - X) - Y_1^0(X_0 - X)}{X_1 - X_0}$$

For the case $K = 3$, the process is repeated six times.

The six iterations give

$$Y_1^1 = \frac{Y_0^0(X_1 - X) - Y_1^0(X_0 - X)}{X_1 - X_0} \quad n = 1, m = 0$$

$$Y_2^1 = \frac{Y_0^0(X_2 - X) - Y_2^0(X_0 - X)}{X_2 - X_0} \quad n = 1, m = 1$$

$$Y_3^1 = \frac{Y_0^0(X_3 - X) - Y_3^0(X_0 - X)}{X_3 - X_0} \quad n = 1, m = 2$$

$$Y_2^2 = \frac{Y_1^1(X_2 - X) - Y_2^1(X_1 - X)}{X_2 - X_1} \quad n = 2, m = 0$$

$$Y_3^2 = \frac{Y_1^1(X_3 - X) - Y_3^1(X_1 - X)}{X_3 - X_1} \quad n = 2, m = 1$$

$$Y_3^3 = \frac{Y_2^2(X_3 - X) - Y_3^2(X_2 - X)}{X_3 - X_2} \quad n = 3, m = 0$$

Note that the fourth and fifth iterations depend upon the first, second, and third, and that the sixth and final one depends on the fourth and fifth. It is evident that the Aitken method may be characterized as the repetitive application of a linear interpolation process.

References

1. Whittaker, E., and Robinson, G.: The Calculus of Observation. Blackie and Sons Ltd., London, 1946.
2. Kopal, Z.: Numerical Analysis. John Wiley and Sons, Inc., New York, 1961.
3. SHARE Program GM TIN2, Table Look-Up and Interpolation.

Identification

AL TAN, Tangent Function
 FORTRAN IV, MAP-coded
 Adapted from the ABC Compiler

Purpose

This subroutine is used to compute the function $Y = \tan(X)$, where X and Y are single-precision floating-point numbers, and X is in radians.

Usage

The subroutine is entered by use of a FORTRAN statement, e.g.,

$Y = \tan(X)$

Special Case

The subroutine returns zero for arguments outside the range -2^{26} to $+2^{26}$, and computation proceeds.

Accuracy

This subroutine gives results accurate to seven significant figures.

Timing

The average computing time for $\tan(X)$ is 456 μ s.

Method

The function $\tan(x)$ is calculated by use of the continued fraction expression:

$$\tan x = \frac{x}{1 + \frac{x^2}{3 + \frac{x^2}{5 + \frac{x^2}{17 + \frac{x^2}{19}}}}}$$

Identification

AL XPAR, Test the Parity of an Integer
FORTRAN IV, MAP-coded
Written by A. L. Summers

Purpose

This subroutine is used to determine the parity of an integer

Usage

The subroutine XPAR is entered by use of the statement

$$J = \text{XPAR}(I)$$

Upon return,

$$\begin{aligned} J &= 0 && \text{if } I \text{ is even} \\ &= 1 && \text{if } I \text{ is odd} \end{aligned}$$
Note

Before its use, XPAR must appear in an INTEGER type statement in the program where it is being used.

Identification

IBM ALG1(ALOG10), ALOG Common and Natural Logarithms

See Writeup for Ames FORTRAN IV Library Routine IBM ALOG

Identification

IBM ALOG, ALG1(ALOG10), Natural and Common Logarithms
FORTRAN IV, MAP-coded
Standard 7090/7094 Library (IBLIB) Routine

Purpose

This subroutine is used to compute common and natural logarithms of a single-precision floating-point positive number.

Usage

This subroutine is entered with the statements

Y = ALOG(X)

and

Y = ALOG10(X)

for natural and common logarithms respectively.

Error Condition

If this subroutine is entered with X equal to zero or a negative value, an error trace is executed, and one of the following two messages is printed.

LOG(0) NOT ALLOWED
LOG(-B) NOT ALLOWED

Control is returned to the calling program. In the case of $X = 0$, the subroutine returns a value of zero for Y, and in the case of entry with negative X, the logarithm of the absolute value is returned.

Accuracy

The absolute error is less than or equal to 3×10^{-8} .

Timing

The execution times for ALOG(X) and ALOG10(X) are approximately 200 μ s and 204 μ s respectively.

MethodA. $\text{Log}_e(X)$ Let $X = 2^I \cdot f$

then

$$\begin{aligned}
 \log_e(X) &= I \log_e 2 + \log_e f \\
 &= I \log_e 2 + \log_2 f \cdot \log_e 2 \\
 &= (I + \log_2 f) \log_e 2
 \end{aligned}$$

The approximation

$$\log_2 f = Z(A + B/(C - Z^2)) - 1/2$$

where

$$Z = \frac{f - \sqrt{2/2}}{f + \sqrt{2/2}}$$

and

$$\begin{aligned}
 A &= 1.2920070987 \\
 B &= 2.6398577035 \\
 C &= 1.6567626301
 \end{aligned}$$

is then employed.

B. $\text{Log}_{10}(X)$

$$\log_{10}(X) = (I + \log_2 f) \log_{10} 2$$

Identification

IBM AND, OR, CMPL(COMPL), BOOL, Boolean Functions
 FORTRAN IV, MAP-coded
 Standard 7094 Library (IBLIB) Routines

Purpose

The built-in functions AND, OR, and COMPL are used to perform the Boolean operations intersection (\cap), union (\cup), and complementation (\sim), with all masking and complementing done on a full 36-bit logical word. The built-in function BOOL is used in the arithmetic IF statement so that a test for zero or non-zero will be performed on the full 36-bit logical word.

Usage

The use of these functions is illustrated in the chart below.

FORTRAN IV Statement	Operation
Y=AND(A,B)	$A \cap B$ (Intersection)
Y=OR(A,B)	$A \cup B$ (Union)
Y=COMPL(A)	$\sim A$ (Complementation)
IF(BOOL(A))100,101,100	If all 36 bits of location A are zero, transfer to statement #101; otherwise go to #100.

The quantities A and B, as well as Y, are full logical words.

Examples

- 1) Let $A = 66_8$ and $B = 71_8$. Then

Y=AND(A,B) yields 60_8

Y=OR(A,B) yields 77_8

Y=COMPL(A) yields 777777777711_8

- 2) Let $A = 400000000000_8$.

Since $A = 400000000000_8$ is the same as $A = -0.0$,

IF(A)100,101,100 will transfer to statement #101 erroneously,

while IF(BOOL(A))100,101,100 will transfer to statement #100 as desired.

Timing

These operations are performed in under 22 μ s.

Comment

- 1) The equivalence between the usage of these functions in FORTRAN IV and Boolean statements in FORTRAN II is shown in the chart below.

FORTRAN IV	*FORTRAN II Boolean Statements
Y=AND(A,B)	Y=A*B
Y=OR(A,B)	Y=A+B
Y=COMPL(A)	Y=-A
IF(BOOL(A))100,101,100	IF(A)100,101,100
* These statements contain a "B" in column 1.	

- 2) A good example of all Boolean operations:

IF(BOOL(OR(AND(A,COMPL(B)), AND (B,COMPL(A)))))100,101,100

Identification

IBM ATAN, ATN2(ATAN2), Inverse Tangent Functions
 FORTRAN IV, MAP-coded
 Standard 7090/7094 Library (IBLIB) Routine

Purpose

This subroutine is used to compute the value of the function $Y = \text{Arctan}(X)$ or $Y = \text{Arctan}(X/Z)$, where X , Y , and Z are single-precision floating-point numbers, and Y is in radians.

Usage

This subroutine is entered with the statements

$$Y = \text{ATAN}(X), \quad -\pi/2 \leq Y \leq +\pi/2$$

or

$$Y = \text{ATAN2}(X, Z), \quad -\pi \leq Y \leq +\pi$$

Error Condition

An error indication is given in the computations using ATAN2 for the case of $X = Z = 0$. An error trace is executed, and then the message

ATAN2(0,0) NOT ALLOWED

is printed. Control is returned to the calling program. The value returned for ATAN2(0,0) is zero.

Accuracy

The absolute error is less than or equal to 10^{-8} .

Timing

The average computing time for ATAN is 278 μ s, and for ATN2, 319 μ s.

Method

A. Arctan(X)

(a) For positive values of X:

(1) If $X > 2^{27}$, $\text{Arctan}(X) = \pi/2$

(2) If $X < 2^{-27}$, $\text{Arctan}(X) = X$

(3) For other values of X, locate X in one of five sub-intervals, according to the location of the corresponding Arctan(X), i.e., $k = 1, 2, 3, 4, 5$, where Arctan(X) is in one of the intervals $0 - 10^\circ$, $10^\circ - 30^\circ$, $30^\circ - 50^\circ$, $50^\circ - 70^\circ$, $70^\circ - 90^\circ$. Then

$$\text{Arctan}(X) = (\pi(k - 1)/9)\text{Arctan } Z_k$$

where

$$\text{Arctan } Z_k = t_k \left[t_k^2 + C_3 - C_2(t_k^2 + C_1)^{-1} \right]^{-1}$$

and

$$t_k = 9X/55 \text{ for } k = 1$$

and

$$t_k = A_k - B_k (X + C_k)^{-1} \text{ for } k = 2, 3, 4, 5$$

and the constants C_1, C_2, C_3, A_k, B_k , and C_k take on appropriate values.

(b) For negative values of X:

$$\text{Arctan}(-X) = -\text{Arctan}(+X)$$

B. Arctan(X/Z)

(a) If $Z = 0$, $\text{Arctan}(X/Z) = \pi/2$ if $X > 0$

$$= -\pi/2 \text{ if } X < 0$$

(b) If $Z \neq 0$, compute $R = X/Z$, and then

(1) If $Z > 0$, $\text{Arctan}(X/Z) = \text{Arctan}(R)$

(2) If $Z < 0$, $\text{Arctan}(X/Z) = \text{Arctan}(R) + \pi$ if $X > 0$
 $= \text{Arctan}(R) - \pi$ if $X < 0$

Identification

IBM ATN2(ATAN2), ATAN, Inverse Tangent Functions

See writeup for FORTRAN IV Library routine IBM ATAN.

Identification

IBM BOOL,AND,OR, CMPL(COMPL), Boolean Functions

See writeup for FORTRAN IV Library Routine IBM AND.

Identification

IBM CMPL(COMPL), BOOL, AND, OR, Boolean Functions

See writeup for FORTRAN IV Library Routine IBM AND.

Identification

IBM COS, SIN, Sine and Cosine Functions

See FORTRAN IV Library Routine IBM SIN.

Identification

IBM DUMP, PDUMP(PDUMP), Selective Dumping Routines
 FORTRAN IV, MAP-coded
 Standard 7090/7094 Library (IBLIB) Routines

Purpose

These dump routines are used to obtain dumps of selected regions of core storage with the option to continue or terminate execution upon completion of the dumping.

Usage

The statements

```
CALL DUMP(A1,B1,F1,...,AN,BN,FN)
```

```
CALL PDUMP(A1,B1,F1,...,AN,BN,FN)
```

are used to enter the subroutines DUMP and PDUMP respectively. The pairs A1-A2,...,AN-BN are variable names that represent upper and lower (or lower and upper) bounds on the regions of core storage to be dumped. The setting of F1,...,FN specifies the dump format for the variables within the given regions, e.g.,

```
F1,...,FN = 0 for octal dump
           = 1 for floating point dump
           = 2 for integer dump
           = 3 for octal dump with mnemonics
```

If no arguments are specified in either of the above CALL statements, then an octal dump of all of core storage is taken. If only F1,...,FN are omitted, then the specified regions are dumped in octal.

Discussion

The two dump procedures are identical in every respect, except that the job is terminated following the execution of the CALL DUMP() statement, whereas computation continues following the execution of the CALL PDUMP statement.

The quantities being dumped are printed out, 8 items per line. The absolute location (in octal) of the first of the 8 is printed to the left of the line.

Identification

IBM EXP, Exponential Function
 7090/7094 FORTRAN IV, MAP-coded
 Standard 7090/7094 Library (IBLIB) Routine

Purpose

This subroutine is used to compute the value of the function $Y = \text{Exp}(X)$, where X and Y are single-precision floating-point numbers.

Usage

This subroutine is entered by use of an arithmetic statement

$$Y = \text{EXP}(X)$$
Error Condition

If this subroutine is entered with a value for X that is greater than 88.028, an error trace will be executed automatically, following which the message

$$\text{EXP}(B), B \text{ GRT TH } 88.028 \text{ NOT ALLOWED}$$

is printed. Control is returned to the calling program. The value returned for Y under the error condition is the input value of X .

Accuracy

The absolute error is less than or equal to 10^{-8} .

Timing

The average execution time for $\text{EXP}(X)$ is 207 μs .

Method

$$\begin{aligned} \text{Exp}(X) &= e^X = 2^{X \log_2 e} \quad (\text{since } e = 2^{\log_2 e}) \\ &= 2^I + F \end{aligned}$$

Method (continued)

where

$$I + F = X \log_2 e$$

and

I = integral part

F = fractional part

Thus

$$e^X = 2^I \cdot 2^F$$

where

$$2^F = 1 + \frac{2F}{D + CF^2 - F - B/(F^2 + A)}$$

and

$$\begin{aligned} A &= 87.417497202 \\ B &= 617.9722695 \\ C &= 0.03465735903 \\ D &= 9.9545957821 \end{aligned}$$

Identification

IBM OR, CMPL(COMPL), BOOL, AND, Boolean Functions

See writeup for FORTRAN IV Library Subroutine IBM AND.

Identification

IBM PDMP(PDUMP), DUMP, Selective Dumping Routines

See writeup for FORTRAN IV Library Routine IBM DUMP.

Identification

IBM SIN, COS, Sine and Cosine Functions
FORTRAN IV, MAP-coded
Standard 7090/7094 Library (IBLIB) Routine

Purpose

This subroutine is used to compute the functions $Y = \sin(X)$ and $Y = \cos(X)$, where X and Y are single-precision floating-point numbers, and X is in radians.

Usage

This subroutine is entered by use of arithmetic statements, e.g.,

$Y = \sin(X)$ for $\sin(X)$,

and

$Y = \cos(X)$ for $\cos(X)$

Error Condition

If this subroutine is entered with an argument that is greater than or equal to 2^{27} in absolute value, an error trace will be executed automatically, following which the message

SIN OR COS ARG GRT TH 2^{27} NOT ALLOWED

is printed. Control is returned to the calling program following this sequence of events. The value returned for $\sin(X)$ or $\cos(X)$ under this condition is zero.

Accuracy

The absolute error is less than or equal to 10^{-8} .

Timing

Average computing times are 247 μ s and 256 μ s for $\sin(X)$ and $\cos(X)$ respectively.

MethodA. $\sin(X)$

$$\text{If } |X| \geq \pi, \sin(X) = (-1)^n \sin(Z)$$

where

$$Z/\pi = X/\pi - n$$

$$\text{and } |Z| < \pi.$$

(a) If $0 \leq |Z| \leq 2^{-8}$, then

$$\sin(Z) = Z$$

(b) If $2^{-8} < |Z| \leq 0.3$, then

$$\sin(Z) = Z(A_1 + 2^{-2} Z^2 + B_1/(Z^2 + C_1))$$

(c) Otherwise, let $m = \pi/2 - |Z|$, and then

$$\sin(Z) = D_1 - 2m^2 - \frac{E - 320m^2}{A_2 + m^2 + \frac{B_2}{m^2 + C_2}}$$

B. $\cos(X) = \sin(X + \pi/2)$

The following constants are used in the above equations:

$$A_1 = -19.8459242619$$

$$B_1 = 1042.92670814$$

$$C_1 = 50.0302454854$$

$$D_1 = 19.47714945237$$

$$A_2 = 82.5803019956$$

$$B_2 = 2287.44319569$$

$$C_2 = 24.1448946943$$

$$E = 3276.33995164$$

Identification

IBM SQRT, Square Root
FORTRAN IV, MAP-coded
Standard 7090/7094 Library (IBLIB) Routine

Purpose

This subroutine is used to compute the square root of a single-precision floating-point non-negative number.

Method

This subroutine is entered by use of the statement

Y = SQRT(X)

Error Condition

If this subroutine is entered with a negative value for X, an error trace is executed, following which the message

SQRT(-B) NOT ALLOWED

is printed. Control is returned to the calling program. The value returned is the square root of the absolute value of the erroneous negative argument.

Accuracy

The absolute error is less than or equal to 10^{-8} .

Timing

The average computing time for SQRT(X) is 142 μ s.

Method

Let $X = 2^{2b} \cdot F$ where $0.25 \leq F \leq 1.0$.

Then

$$\sqrt{X} = 2^b \sqrt{F}$$

where

$$\sqrt{F} \approx P_i \quad (i \text{ is the number of the approximation})$$

Now, the first approximation is

$$P_1 = AF + B$$

where

$$A = 0.875, \text{ and } B = 0.27863 \text{ for } 0.25 \leq F < 0.5$$

and,

$$A = 0.578125 \text{ and } B = 0.421875 \text{ for } 0.5 \leq F \leq 1$$

Then two Newton iterations are carried out:

$$P_2 = (P_1 + F/P_1)/2$$

$$P_3 = (P_2 + F/P_2)/2$$

The final result is the iteration P_3 .

Identification

IBM TANH, Hyperbolic Tangent Function
 FORTRAN IV, MAP-coded
 Standard 7090/7094 Library (IBLIB) Routine

Purpose

This subroutine is used to compute the function $Y = \tanh(X)$, where X and Y are single-precision floating-point numbers.

Usage

This subroutine is entered with the statement

$$Y = \text{TANH}(X)$$
Accuracy

The absolute error is less than or equal to 3×10^{-8} for X in the range .0034-.17, and less than 10^{-8} elsewhere.

Timing

The average computing time for TANH(X) is 303 μ s.

Method

A. For positive X :

(a) For $|X| \geq 12$

$$\tanh(X) = 1$$

(b) For $12 > X \geq .17$

$$\tanh(X) = \frac{e^X - e^{-X}}{e^X + e^{-X}} = \frac{e^{2X} - 1}{e^{2X} + 1}$$

(c) For $.17 > X > .00034$

Method (continued)

$$\tanh(X) = f(A + f^2(B + C(D + f^2)^{-1}))^{-1}$$

where

$$f = 4X \log_2 e$$

and

$$\begin{aligned} A &= 5.7707801636 \\ B &= 0.01732867951 \\ C &= 14.1384114018 \\ D &= 349.6699888 \end{aligned}$$

(d) For $.00034 \geq X$

$$\tanh(X) = X$$

b. For negative X:

$$\tanh(-X) = -\tanh(+X)$$

Identification

ML HPRS, AL DPMU, Real or Complex Roots of a Polynomial with Real Coefficients
FORTRAN IV

Ames Modifications of SHARE Library Routine ML HPRS

Purpose

These routines are used to calculate real or complex roots of a polynomial with real coefficients. The subroutine ML HPRS uses single-precision arithmetic, and AL DPMU uses double-precision arithmetic and coefficients and roots are double-precision numbers.

Usage

The subroutines are called by use of the statements

```
CALL MULLER(COE, N, ROOTR, ROOTI)
```

for the single-precision routine and

```
CALL DPMUL(COE, N, ROOTR, ROOTI)
```

For the double-precision routine, where

COE is the ~~name~~ of the array where the coefficients of the polynomial are stored, ordered from the highest degree to the lowest degree

N is the degree of the polynomial

ROOTR is the name of the array where the real parts of complex roots are to be stored

ROOTI is the name of the array where the imaginary parts of complex roots are to be stored

The arrays COE, ROOTR, and ROOTI must be named in a DOUBLE PRECISION type statement in the use of the double-precision subroutine DPMU.

All arithmetic in these routines is in the complex mode. Therefore, all roots will have an imaginary part as well as a real part, and the "imaginary part" of a real root will be smaller by about seven decimal places than the real part.

Accuracy

A nonmultiple root is generally accurate to from six to eight decimal places. Multiple roots are computed less accurately, however, and when the multiplicity is four the accuracy is only to about two decimal places.

Method

A method due to Muller [MTAC (1959) pp. 208-215] is used in these routines.